UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sami Ilc

Optimizacija skladiščnih procesov s pomočjo mobilne tehnologije

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

MENTOR: doc. dr. Damjan Vavpotič

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomske naloge razvijte delujoč prototip sistema, ki bo omogočil optimizacijo izbranih skladiščnih procesov. Funkcionalnosti prototipa naj po eni strani temeljijo na predhodni analizi obstoječih podobnih sistemov, po drugi strani pa se še posebej osredotočite na dejanske potrebe izbranega podjetja, v katerem boste prototip preizkusili. Prototip mora vsebovati vse potrebne funkcije in omogočati povezavo z zalednim sistemom. V diplomski nalogi predstavite ključne funkcionalnosti, programsko arhitekturo in delovanje prototipa. Pomemben del diplomskega dela bo preizkus prototipa na realnem primeru, kjer boste še posebno pozornost namenili merjenju učinkovitosti izdelanega prototipa v realnih okoliščinah. Pripravljen prototip kritično ovrednotite in podajte smernice za nadaljnje delo.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Sami Ilc, vpisna številka 63080064, avtor zaključnega dela z naslovom:

Optimizacija skladiščnih procesov s pomočjo mobilne tehnologije
*(*angl. *Optimization of warehouse processes by using mobile technology)*

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom doc. dr. Damjana Vavpotiča;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 29. julija 2016 Podpis študenta/-ke:

Neizmerna zahvala gre mojim staršem za vsestransko podporo čez celoten študij, mentorju doc. dr. Damjanu Vavpotiču, za vztrajno in korektno sodelovanje, mojemu dekletu Anji, ki me je nenehno vzpodbujala in prenašala tudi v težkih trenutkih, prijatelju Maticu ter celotni družini za koristne nasvete in pomoč ter vsem sodelujočim pri raziskavi, brez katerih diplomskega dela ne bi bilo mogoče realno ovrednotiti.

Kazalo

Seznam uporabljenih kratic

Kratica	Angleško	Slovensko
API	application program interface	programski vmesnik
CSS	cascading style sheets	stilna predloga na spletni strani, v kateri je zapisana oblika spletne strani
CSV	comma-seperated value	običajni format za besedilno datoteko, ki vsebuje z vejico ločene vrednosti
ERP	enterprise resource planning	celovita programska rešitev
GCM	google cloud messaging	googlovo oblačno sporočanje
GSON	google script object notation	googlov zapis skriptnih objektov
HTTP	hypertext transfer protocol	protokol za izmenjavo hiperteksta ter grafičnih, zvočnih in drugih večpredstavnostnih vsebin na spletu
JS	java script	splošno uporaben, skriptni programski jezik, sintaktično podoben javi
JSON	javascript object notation	zapis javascript objektov
OCR	optical character recognition	optično prepoznavanje znakov
ODBC	open database connectivity	standardna SQL-metoda pristopa iz poljubne aplikacije do podatkovnih baz
OS	operating system	operacijski sistem
PDF	portable document format	format datoteke, ki je neodvisen od računalniškega okolja in medoperacijsko prenosljiv
PHP	PHP: hypertext preprocessor	splošno uporaben skriptni programski jezik, ki ga tolmači strežnik
RFID	radio-frequency identification	radiofrekvenčno prepoznavanje

RPC	remote procedure call	klic za oddaljeni postopek
SCM	supply chain management	upravljanje oskrbovalne verige
SHA	secure hash algorithm	kriptografska zgoščevalna funkcija
SQL	structured query language	strukturiran povpraševalni jezik za delo s podatkovnimi bazami
UI	user interface	uporabniški vmesnik
UML	unified modeling language	poenoteni jezik modeliranja
URL	uniform resource locator	enotni naslov vira
XML	extensible markup language	format podatkov za izmenjavo strukturiranih dokumentov v spletu

Povzetek

Naslov: Optimizacija skladiščnih procesov s pomočjo mobilne tehnologije

Diplomsko delo z uvodnimi besedami izpostavi problematiko razvoja poslovnih mobilnih aplikacij, nadaljuje pa z motivom raziskave o uporabnosti mobilne tehnologije pri postopkih skladiščenja sodelujočega podjetja. Najprej nas seznani s teorijo skladiščnih logistik, ki se dandanes izvajajo v praksi, ter se dotakne njihovih problematik. Temu sledi podrobnejša analiza trenutne izvedbe skladiščnih procesov obravnavanega okolja ter ob njem uporabljene programske opreme. Delo poudari kritične točke skladiščenja ter opiše potrebne osnovne funkcionalnosti informacijskega sistema, s katerimi bi lahko odpravili določene časovne zamude. Nato se preverijo morebitne obstoječe mobilne rešitve na trgu, ki bi vsebovale zahtevane funkcije, ter smiselnost njihove dejanske uporabe. Osrednji del naloge celovito opiše razvoj novega prototipnega mobilnega sistema, postopek njegove implementacije v realno skladiščno okolje, ter ob tem nastale težave. Sledi epilog s predstavitvijo ter končno analizo pridobljenih rezultatov testiranja razvitega prototipa. Delo se zaključi s podanim strokovnim mnenjem vodstva podjetja o nastalem izdelku ter končnim sklepom o njegovi dodani vrednosti.

Ključne besede: skladiščenje, skladiščni procesi, mobilni informacijski sistem

Abstract

Title: Optimization of warehouse processes by using mobile technology

Introductory of this dissertation highlights the problems of business mobile applications development, it continues with the motive of research on the usefulness of mobile technologies in warehouse processes of company involved. First of all, it introduces us with the theory of warehouse logistics, which are nowadays carried out in practice, and touches their issues. This is followed by a detailed analysis of the current implementation of warehouse processes of the environment involved and software used for it. It emphasizes the critical points of storage and describes the necessary basic functionality of the information system, which could eliminate certain delays. After that, follows the check for any existing mobile solutions on the market, which would include the required function and viability of their actual use. The main part of this dissertation comprehensively describes the development of a new prototype of a mobile system, the process of its implementation in the real warehouse environment, and difficulties encountered. Epilogue contains a presentation and final analysis of the results of testing the developed prototype. The dissertation concludes with a given expert opinion of company management for the resulting product and the final decision on its added value.

Keywords: storage, warehouse processes, mobile information system

Poglavje 1 Uvod

V današnjem svetu sta tehnološki napredek in razvoj industrije popolnoma spremenila naš način življenja. Čas je postal neprecenljiva vrednota. Vsakodnevna opravila, zasebna ali službena, vedno skušamo narediti hitreje, iščemo krajše poti do njihovih zaključkov, da bi le prihranili čim več časa in povečevali lastno storilnost. Ena izmed panog, ki bistveno pripomore k temu, je nedvomno mobilna tehnologija.

Njena ključna vrednost je omogočanje prenosljive komunikacije, v zadnjem desetletju pa doživlja pravi razcvet. Ob začetku tega tisočletja pa vse do danes je standardna mobilna naprava iz preprostega dvosmernega pozivnika postala mobilni telefon, navigacijski sistem GPS, vgrajeni spletni brskalnik, dinamičen sporočilni sistem, video igralni sistem in še veliko več. Mnogi strokovnjaki trdijo, da prihodnost računalništva leži v mobilni tehnologiji z brezžičnim omrežjem. Ta sicer postaja vse bolj priljubljena v obliki tabličnih računalnikov in pametnih telefonov. [1]

Zgornjo trditev potrjuje tudi statistika. Ob koncu prejšnjega leta je bilo registriranih preko 4,7 milijarde posameznih uporabnikov mobilnih telefonov, kar je več kot 60 odstotkov celotne svetovne populacije. Zaradi vse večje cenovne dostopnosti pa se povečuje tudi delež imetnikov pametnih naprav. Do leta 2020 naj bi ta znotraj držav v razvoju znašal dve tretjini vseh uporabnikov, v razvitih pa kar tri četrtine. [2]

Širitev uporabe pametne telefonije vpliva tudi na spletni trg, ki se čedalje bolj usmerja v razvoj različnih mobilnih aplikacij, odprtokodni OS določenih ponudnikov, nizki stroški implementacije pa samostojnim razvijalcem odpirajo vrata za nove poslovne priložnosti. S tem nastaja visoka konkurenčnost. Tržišče aplikacij za zasebne uporabnike je danes nasičeno z najrazličnejšimi produkti, od spletnih trgovin, socialnih omrežij, računalniških iger, sporočilnih, navigacijskih sistemov in še več. Na drugi strani pa je učinkovitejša uporaba mobilnih naprav v poslovnem svetu še precej neraziskana. Eden izmed razlogov temu je, da se za uspešno integracijo takšnega produkta od razvijalcev zahteva temeljito poznavanje procesov problematičnega okolja znotraj družb, kar pa večini ni omogočeno.

V okviru diplomskega dela se je ponudila priložnost sodelovanja s podjetjem, ki v Sloveniji zaposluje več kot 3000 ljudi in je eno izmed vodilnih v svoji panogi. Zaradi izražene želje po

anonimnosti družba ne bo imenovana, prav tako so v nadaljevanju uporabljene informacije in podatki, ki bi lahko razkrili njeno identiteto, ustrezno cenzurirani oziroma prilagojeni.

Primarni motiv naloge je tako bil raziskati potencialno uporabnost mobilne tehnologije v realnem poslovnem okolju. Ker se podjetje srečuje z informacijskimi težavami ter oteženo dostopnostjo do računalniške opreme ob izvedbi postopkov skladiščenja izdelkov, se je porodila ideja, da izdelamo testni mobilni sistem, ki bi te procese lahko optimiziral. Pred razvojem in dejansko implementacijo takšnega prototipa pa je najprej bilo potrebno celovito razumevanje skladiščne logistike ter podrobnejša analiza samih postopkov in uporabljene tehnologije znotraj družbe. Slednji nas bosta seznanili z dejansko problematiko skladiščenja in zakaj do nje sploh prihaja. Temu je sledila preverba morebitnih obstoječih rešitev na trgu ter njihova uporabnost glede na izpostavljen problem. Glavni prispevki naloge so razvoj delujočega prototipa sistema, njegova postavitev v obstoječe skladiščno okolje ter končni rezultati testiranja njegove uporabnosti. Ta je ob zaključku diplome ovrednotena s podanim strokovnim mnenjem vodstva, ki je tudi opredelilo možnost dejanske produkcijske implementacije izdelanega prototipa.

Poglavje 2 Teorija skladiščne logistike

Razvit prototip v okviru diplomskega dela rešuje eno izmed problematik skladiščnih procesov, zato je smiselno, da se bralca najprej seznani z osnovami logistike proizvodnih podjetij, kakršno je tudi sodelovalo pri raziskavi. V poglavju je predstavljeno, kdo, kaj in kako z njo upravlja ter kakšne strategije so uporabljene pri njeni implementaciji, ki bistveno vplivajo na sam potek skladiščenja izdelkov.

2.1 Oskrbovalna veriga

Sistem, ki skrbi za tekoče delovanje podjetne logistike, se imenuje oskrbovalna veriga (v angl. »supply chain«). Je mreža proizvajalcev in ponudnikov storitev, ki delujejo skupaj za preoblikovanje in premik blaga od faze surovin pa vse do končnega kupca. Ti so med seboj povezani s fizičnimi, informacijskimi in denarnimi tokovi. Oskrbovalna veriga obstaja v storitvenih in proizvodnih podjetjih, vendar se kompleksnost verige lahko zelo razlikuje med različnimi panogami. Ključna cilja njenega upravljanja (v angl. »Supply Chain Management« – SCM) sta učinkovita izvedba vseh dejavnosti v verigi ter vzpostavitev odnosov znotraj nje z namenom, da poveča vrednote strank in doseže trajnostno konkurenčno prednost. [3]

Glavne odločitve uprave SCM obsegajo štiri glavna področja:

1. Lokacije objektov (proizvodnih, skladiščnih ter objektov nabave).

2. Proizvodnjo, ki vključuje odločitve o tem, kaj proizvesti in v katerih obratih, ter o alokaciji dobaviteljev v posamičnih obratih.

3. Zaloge, kar se nanaša na način upravljanja z njimi. Nastajajo v vsaki fazi oskrbovalne verige, in sicer kot:

    - surovine (naravni materiali, nedokončane dobrine, ki običajno niso uporabne, dokler jih proizvajalci ne predelajo v končne proizvode),

    - sestavni deli. (končani elementi, ki se uporabljajo v zadnjih korakih obdelave končnih produktov),

- pol-izdelki (surovine ali sestavni deli, ki so že v procesu izdelave in so v čakanju na sprostitev sredstev, da se predelajo v končno obliko) ter

- končni izdelki (kupljeni proizvodi, naprave ali storitveni deli, po katerih povpraševanje prihaja z naročili strank ali napovedi prodaje). [5]

4. Transport, kar je strateškega pomena. Odločitve transporta so tesno povezane z odločitvami o zalogah, saj so stroški obeh v indirektni povezavi. [4]

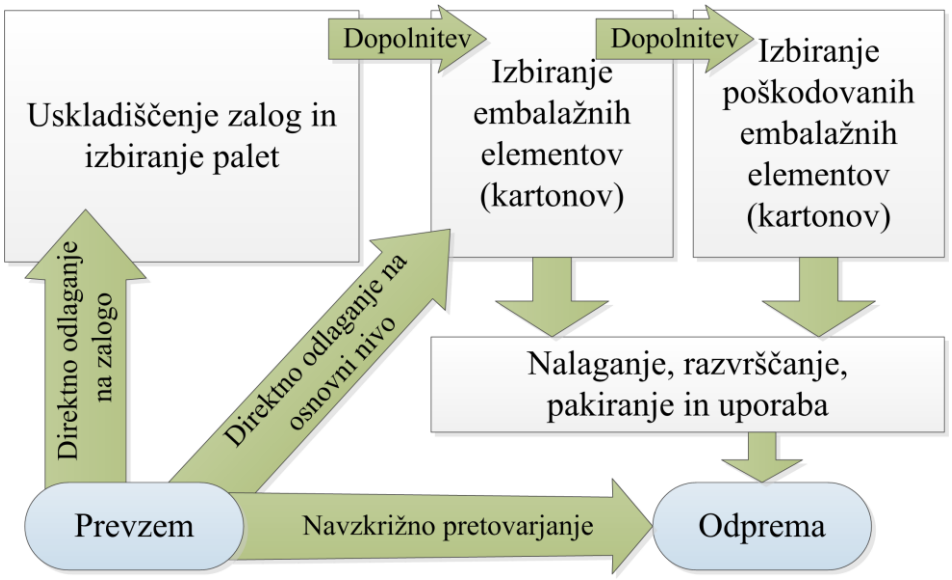
V nadaljevanju smo se osredotočili predvsem na opredelitev operacije skladiščenja, kar je tudi osrednja tematika naloge.

2.2 Skladiščenje izdelkov

Skladiščenje je funkcija oz. del oskrbovalne verige, ki skrbi za fizično shranjevanje vseh vrst blaga na zalogi v času od točke izdelave pa vse do odjemnega mesta, kjer mora biti blago dostavljeno. Prav tako vodstvu razpolaga z informacijo o stanju, pogojih in razporeditvi izdelkov, ki se trenutno skladiščijo. [6]

2.2.1 Skladiščni procesi

Znotraj skladiščenja se vršijo tri osnovne operacije: gibanje blaga, shranjevanje ter prenos informacij. Prva je v zadnjem času deležna največje pozornosti, saj se uprava SCM čedalje bolj osredotoča na izboljšavo obračanja zalog in pospešitev naročil vse od proizvodnje pa do končne dostave. In ker je med drugim v nadaljevanju predstavljena tudi optimizacija postopkov znotraj funkcije gibanja, je dobro poznati njene interne procese, ki so podrobneje prikazani na sliki 2.1.



Slika 2.1: Tipični skladiščni procesi in tokovi

Prevzem je skladiščni proces, ki vključuje dejansko razkladanje izdelkov iz prevoznega sredstva, posodobitev evidence zalog, pregled poškodb in potrjevanje prejete količine glede na naročilo. Tok odlaganja oziroma prenosov fizično premika prejeto blago znotraj skladišča. Ta gre lahko iz prevzemne točke neposredno na skladiščenje, na območje za specializirane storitve, kjer se konsolidira ali pa postavi na izhodno pošiljanje. Navzkrižno pretovarjanje (v angl. »cross-docking«) zaobide omenjene skladiščne dejavnosti in gre direktno na odpremo, zadnjo gibalno aktivnost. Znotraj slednje se preverijo sestavljena naročila, ki morajo biti poslana. Fizično jih selijo na opremo prevoznika ter prilagodijo evidenco zalog. Izdelki so nameščeni v škatle, kartone ali druge zabojnike, označene s potrebnimi informacijami za pošiljko (poreklo, pošiljatelj, prejemnik, destinacija, vsebina paketa) in položene na paleto. [6]

2.2.2 Načini skladiščenja

Način skladiščenja je za podjetje izrednega strateškega pomena, saj vpliva na celotno vzpostavljeno logistiko. Cilj družbe je izbrati tistega, ki zagotavlja možnost, da se dosežejo najmanjši skupni stroški, potrebni za uspešno in učinkovito izvedbo logističnih funkcij, obenem pa olajšati doseganje zastavljenih ciljev storilnosti podjetja.

Poznamo naslednje osnovne oblike skladiščenja:

- zasebno,
- javno oz. najemno ter njegovi podvrsti:
    - pogodbeno in
    - prehodno (v angl. »in-transit«).

Zasebno skladiščenje se od ostalih oblik razlikuje v tem, da so lastnina, objekti, spremljajoče shranjevanje blaga ter celotna industrijska oprema v lasti podjetja, ki jih hkrati tudi upravlja. Omenjena oblika je lahko sestavljena zgolj iz enega regala ali skladiščne sobe, pa vse do kompleksne mreže več tisoč kvadratnih metrov velikih skladišč, razpostavljenih po različnih celinah. Javno oz. najemno skladiščenje, na drugi strani, dandanes predstavlja stalnico fizičnih distribucijskih strategij številnih družb. Vse večji pomen javnega skladiščenja lahko zasledimo v spreminjajoči se naravi svetovnega trga. S celovito pokritostjo shranjevanja izdelkov, njihovega upravljanja, pisarniških storitev ter storitev z dodano vrednostjo, javno skladiščenje zagotavlja kratke in dolgoročne funkcije, namenjene podpori zahtevam strank oskrbovalne verige. Ključna razlika najemnega skladišča v primerjavi z zasebnim je, da so objekti in industrijska oprema v lasti drugega podjetja, ki storitve skladiščenja zaračunava v obliki mesečne pristojbine. Če gre za dolgoročno pogodbo med družbama, govorimo o pogodbenem skladiščenju. Cilj slednjega je vzpostaviti jamstvo s strani podjetja, da bo nivo poslovanja v času pogodbenega roka nespremenljiv, hkrati pa daje obema stranema priložnost, da skupaj poiščejo nove načine za zmanjševanje stroškov ter poskrbijo za vzpostavitev boljše komunikacije in opravljanja storitev. Prehodno oz. tranzitno skladiščenje pa predstavlja shranjevanje izdelkov v času transporta (na letalu, ladji, vlaku, tovornih vozilih, …). [5, 6]

Da podjetje smiselno ubere zanj ustrezno logistično strategijo, mora biti seznanjeno s ključnimi prednostmi in slabostmi tako javnega kot zasebnega načina skladiščenja, ki so predstavljene v spodnji tabeli 2.1.

Prednosti zasebnega skladiščenja:	**Prednosti javnega skladiščenja:**
1. neposreden nadzor skladiščnih operacij in procesov (pomemben, kadar podjetje proizvaja izdelke (npr. farmacevtske), ki zahtevajo posebne pogoje shranjevanja)	1. naložba v dolgoročna sredstva (ne zahteva fiksnih naložb, stroški so spremenljivi in sorazmerni glede na porabo skladiščnih storitev, **investicija v razvoj skladišč ni potrebna**)
2. **ni težav s komunikacijo** (neposreden stik z upravo, sistemi in terminali so združljivi z informacijskim sistemom podjetja)	2. zmanjšanje tveganja zastarelosti objektov in uporabljene tehnologije (sledijo trendom na trgu)
3. fleksibilnost pri načrtovanju in upravljanju skladišča (z neposrednim nadzorom je omogočena večja stopnja prilagodljivosti glede na potrebe strank in značilnosti produktov)	3. fleksibilnost lokacije (velika prednost, kadar poslovni pogoji zahtevajo spremembo skladiščnih lokacij)
4. možnost davčnih ugodnosti zaradi lastninjenja nepremičnin (podjetje daje skupnosti in strankam občutek zavezanosti in stalnosti)	4. davčne olajšave (izogibanje lokalnim davkom, ki izhajajo iz lastništva nepremičnin, prav tako nekatere države ne zaračunavajo davka na zaloge, shranjene v skladišču)
5. cenejše, kadar je za poslovne procese podjetja značilna visoka izkoriščenost skladiščnega območja (vsaj 75–80 odstotkov) ter ob večjih, stalnih količinah blaga	5. poznavanje točnih stroškov skladiščenja in upravljanja (prejemanje mesečnih računov, stroški predvideni vnaprej)

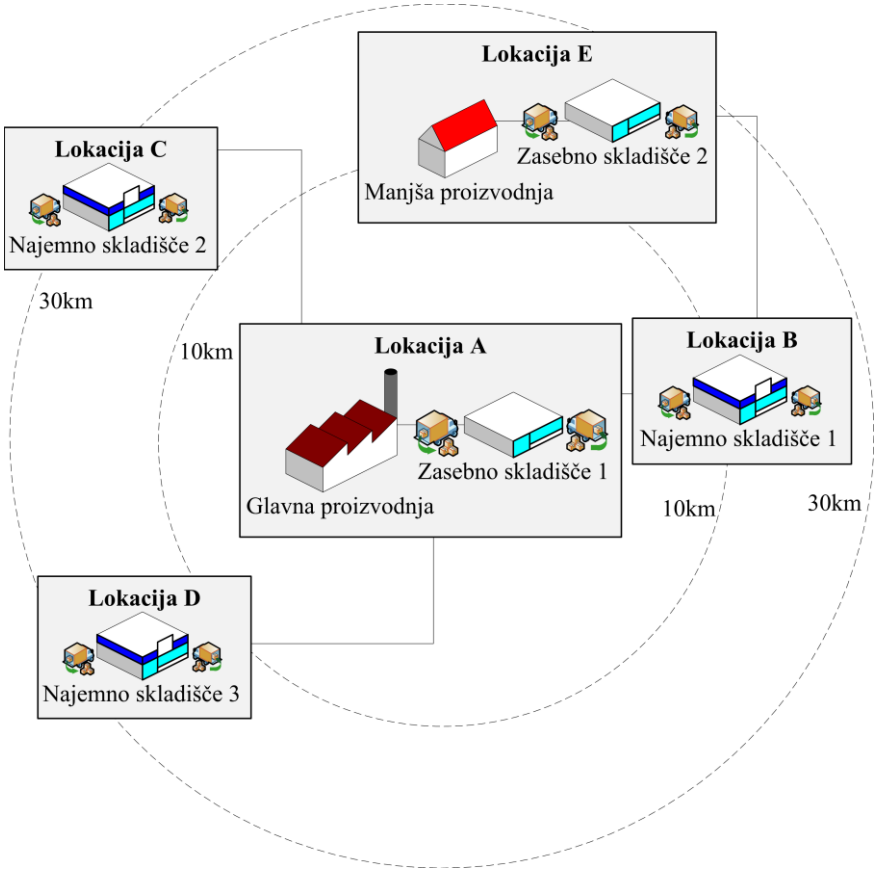
Tabela 2.1: Primerjava prednosti javnega in zasebnega skladiščenja [5, 6]

Ker prednosti enega načina skladiščenja predstavljajo slabosti drugega, slednje niso bile posebej izpostavljene. Kot lahko vidimo, ima vsaka vrsta svoje strateške prednosti.

Dejavniki, ki vplivajo na končno odločitev SCM, so vrsta industrije, njegovi cilji, finančne sposobnosti, lastnosti izdelkov (pokvarljivost, količina, potencial zastarelosti, moč konkurence, …) ter splošno stanje gospodarstva v državi. [5, 6]

2.3 Problematika skladiščenja sodelujočega podjetja

Podjetje, ki je sodelovalo pri raziskavi, večji del blaga, predvsem končne izdelke, hrani v najemnih skladiščih oz. depojih, s katerimi ima sklenjeno dolgoročno pogodbo, torej uporablja pogodbeni način javnega skladiščenja. V lasti pa ima tudi zasebna, ki se nahajajo na isti lokaciji kot proizvodni obrati (slika 2.2).



Slika 2.2: Shema skladiščnih lokacij sodelujočega podjetja

Ob analizi postavljenega logističnega okolja družbe je prišlo do zanimivih ugotovitev. Izkazalo se je, da je razporeditev prostorov in opreme v zasebnih depojih res bolje načrtovana od najemnih, kar potrjuje tretjo točko podanih prednosti privatnega tipa. Razlog je v tem, da podjetje zaradi sklenjene časovno omejene pogodbe nima ustreznih dovoljenj za poseganje v infrastrukturo javnih skladišč, prav tako mu ni v lastnem interesu, saj se mu na dolgi rok ne izplača. Slaba dostopnost do potrebnih naprav in računalnikov ob izvajanju skladiščnih operacij pa zaposlenim lahko povzroča neželene časovne izgube.

Druga slabost najetih objektov je tudi v slabši avtomatizaciji. Namreč najemnik, iz istih razlogov kot zgoraj, ne investira v njihov tehnološki razvoj, kar je sicer v tabeli 2.1 s prvo točko podano kot prednost javnega skladiščenja, a predvsem zaradi potencialnih prihrankov. Posledično se uporabljajo terminali, ki so na voljo. Čeprav najemodajalci načeloma skrbijo za moderniziranost svojih skladišč, imajo ta vzpostavljene informacijske sisteme in naprave, ki niso nujno združljivi s sistemom in logistiko podjetja, ki jih najema, kar pa negativno vpliva na sam potek skladiščnih procesov.

Tretji problem je komunikacija. Medtem ko v zasebnih depojih večjih logističnih težav ni, saj skladiščniki vso potrebno informacijo o blagu dobijo neposredno iz proizvodnje in v primeru odstopov pravočasno ukrepajo, v javnih skladiščih, ki so stacionirana na različnih krajih, stran od proizvodnega obrata, kot prikazano na sliki 2.2, prihaja do neskladij. Večkrat se zgodi, da pride v pakirnici do spremembe pakiranja izdelkov, bodisi zaradi uporabe nove embalaže, optimalnejšega načina pošiljanja paketov z manjšimi oz. večjimi količinami izdelkov ali drugih razlogov, o katerih oskrba ni obveščena, ker ne vpliva na zahtevana naročila strank. Ob prevzemu blaga morajo skladiščniki preveriti skladnost prejetih artiklov s podatki v sistemu, ki pa niso ustrezno posodobljeni. Ker pride v tem primeru do odstopanj, so potrebni popravki v podatkovni bazi in dodatne uskladitve z oskrbo, kar lahko postane zamudno. Težava je še večja, če se napake odkrijejo šele pri postopku odpreme, saj morajo izdelki čimprej iz skladišča, da se sprostijo kapacitete za naslednje prispele pošiljke in da izstopno blago pravočasno prispe do kupca. To se lahko pojavi npr. pri navzkrižnem pretovarjanju. Posledično lahko nastanejo dodatni nepredvideni stroški, neugodni za družbo.

Učinkovitejše komuniciranje se lahko znotraj podjetja rešuje na različne načine. A pri tem nastane večji problem, ko je potrebno skladiščiti izdelke, ki prihajajo od zunanjih dobaviteljev oz. tretjih strank (v angl. »third party«). Običajno so to surovine ali pol-izdelki, ki se nato predelajo v končne produkte, ali pa kar končni izdelki, ki se kupijo in nato preprodajo. Če pri lastnih proizvodih podjetja pride do odstopov zgolj pri nenadnih spremembah pakiranj, je pri tujem blagu možnost neskladij veliko večja. Namreč, v veliko primerih se dogaja, da strani uporabljata različne informacijske sisteme, standarde ter način zagotavljanja pakirnih podatkov. Uprava SCM je sicer seznanjena s kupljeno vsebino, količino, izvorom in časom prejetih artiklov, ki so sestavni del naročila, medtem ko je način pakiranja izdelkov v embalažne elemente in zlaganja na paleto znan šele ob prihodu blaga v lokalno skladišče. To za dobavitelja namreč ni kritična informacija in jo poda zgolj pri vsebini dobavnice ali z njo sploh ne razpolaga. Politika sodelujočega podjetja pa zahteva, da mora imeti vsako skladiščeno blago v sistemu ustrezno izpolnjene pakirne podatke, saj le ti omogočajo sistemski nadzor oskrbe nad skladiščenjem, s tem pa zagotavljajo kakovostnejšo predajo naročil končnemu kupcu. Seveda pa omenjeni postopek za seboj potegne dodatno delo za skladiščnike, ki morajo preveriti skladnost sistema s prejetimi artikli. V primeru napak morajo oskrbi posredovati podatke, pridobljene iz dobavnic. V primeru, da pakirni podatki manjkajo, jih ročno izmerijo, da jih lahko oskrba posodobi v podatkovni bazi. Posledično lahko nastanejo časovne zamude pri odpremah, hkrati pa so skladiščniki s tem še dodatno obremenjeni.

Če povzamemo, ključne težave pri izvedbi skladiščnih procesov, s katerimi se srečujemo predvsem v najemnih depojih, obsegajo: dostopnost skladiščne opreme, združljivost skladiščne opreme z obstoječo logistično opremo podjetja ter težave pri komunikaciji. Ker družba posluje

večinoma z javnimi skladišči, večja investicija v sodobnejšo tehnologijo in boljšo infrastrukturo ni upravičena oz. mogoča. Prav tako je možnost pridobitve bogatejših informacij o načinu pakiranja tujih izdelkov s strani dobaviteljev pred samim prihodom blaga v skladišče težko izvedljiva, saj blago dobavljajo podjetja iz različnih držav, kar pomeni, da bi bilo potrebno vložiti ogromno truda za standardizacijo pakirnih podatkov, poleg tega pa dobavitelji niso stalni.

Rešitev bi torej bila implementirati tehnologijo z naslednjimi lastnostmi:

- združljivost s katerim koli skladiščnim okoljem, kar bi rešilo problem javnega skladiščenja,
- prenosljivost uporabljenih naprav, ki bi odpravile težave z dostopnostjo,
- nizki stroški vzpostavitve in vzdrževanja,
- enostavnost uporabe ter
- hitra integracija v obstoječi sistem.

Ena izmed sodobnejših in razvijajočih se tehnologij današnjega časa, ki bi ustrezala vsem zgoraj naštetim pogojem, je mobilna tehnologija, ki je hkrati tudi uporabljena rešitev v okviru diplomske naloge.

2.4 Obstoječa mobilna tehnologija v današnjih skladiščih

V današnjih skladiščih se mobilna tehnologija že aktivno uporablja, a večinoma zgolj za identifikacijo skladiščenih izdelkov. Prepoznava blaga je ključna pri prevzemu in odpremi, saj z njo lahko sistemsko beležimo stanje zalog v depojih, hkrati pa končnemu kupcu omogoča sledenje naročenih artiklov. Deluje tako, da se na vsak skladiščeni element, lahko je samo karton ali pa cela paleta, namesti ustrezna identifikacijska oznaka. Ta je lahko v obliki črtne kode ali pa oznake RFID (»radio frequency identification«), odvisno od implementacije. Prva velja za bistveno cenejšo rešitev, saj se jo lahko natisne na navaden list papirja ali neposredno na embalažo artikla, medtem ko je za RFID potrebna vgradnja računalniškega čipa, ki pa se izkaže za učinkovitejšo rešitev pri dejanski prepoznavi. Obe vrsti oznak se prebereta s posebnimi čitalci, ki delujejo skupaj z mobilnimi napravami. Te so preko brezžičnega omrežja povezane z informacijskim sistemom podjetja in iz njegove podatkovne baze črpajo ali pošiljajo identifikacijske podatke o skladiščenem blagu, ki se nato izpišejo na grafičnih vmesnikih naprav.

Velikokrat pa se dogaja, da dobavitelji zunanjih izdelkov ne razpolagajo z ustreznimi oznakami na prejeti embalaži. V takšnih primerih morajo skladiščniki za uspešno prepoznavo blaga ob prevzemu na posamezen element, še preden ga uskladiščijo, najprej ročno namestiti novo, ustrezno identifikacijsko oznako, kar je spet zamudno. Prav tako opisana tehnologija v primeru sistemskih neskladij ni zmožna posredovati podrobnejših pakirnih podatkov, kot so način zlaganja znotraj kartona in na paleto, dimenzije, teža posameznega embalažnega elementa in drugi, s katerimi oskrba svojim strankam želi zagotoviti kakovostnejšo predajo naročenih artiklov. Z informacijo o identiteti, ki jo dobimo s čitalci, lahko zgolj nadziramo količine prejetega in odposlanega blaga ter poizvemo, kje se blago trenutno nahaja. Torej bi za omenjeno problematiko potrebovali drug vmesnik, ki bi ponujal možnost pošiljanja celovitih pakirnih podatkov, oz. bi morali obstoječega nadgraditi. To pa lahko predstavlja velike stroške, saj so posegi v obstoječe sisteme razmeroma dragi.

Smiselna rešitev bi lahko bila mobilna aplikacija na pametnih telefonih, ki bi delovala na platformi, ločeni od obstoječe, kar bi bilo glede na celotno implementacijo cenovno ugodnejše in tudi lažje izvedljivo, hkrati pa bi omogočala prilagoditev potrebnih vnosov glede na potrebe poslovnega okolja. Aplikacija bi tudi morala imeti možnost pošiljanja pakirnih podatkov oskrbi v ustreznem zapisu, s katerim bi se lokalna podatkovna baza obstoječega sistema hitro in enostavno posodobila. Številna podjetja, vključno s sodelujočim, svojim zaposlenim že priskrbijo službene pametne telefone, a se ti uporabljajo večinoma zgolj za telefoniranje, pregledovanje elektronske pošte ali pošiljanje SMS sporočil in ne izkoriščajo potencialne dodane vrednosti, ki jih naprava ponuja. Omenjena rešitev bi torej bila ugodna tudi zato, ker ni potrebe po dodatnem nakupu tehnološke opreme.

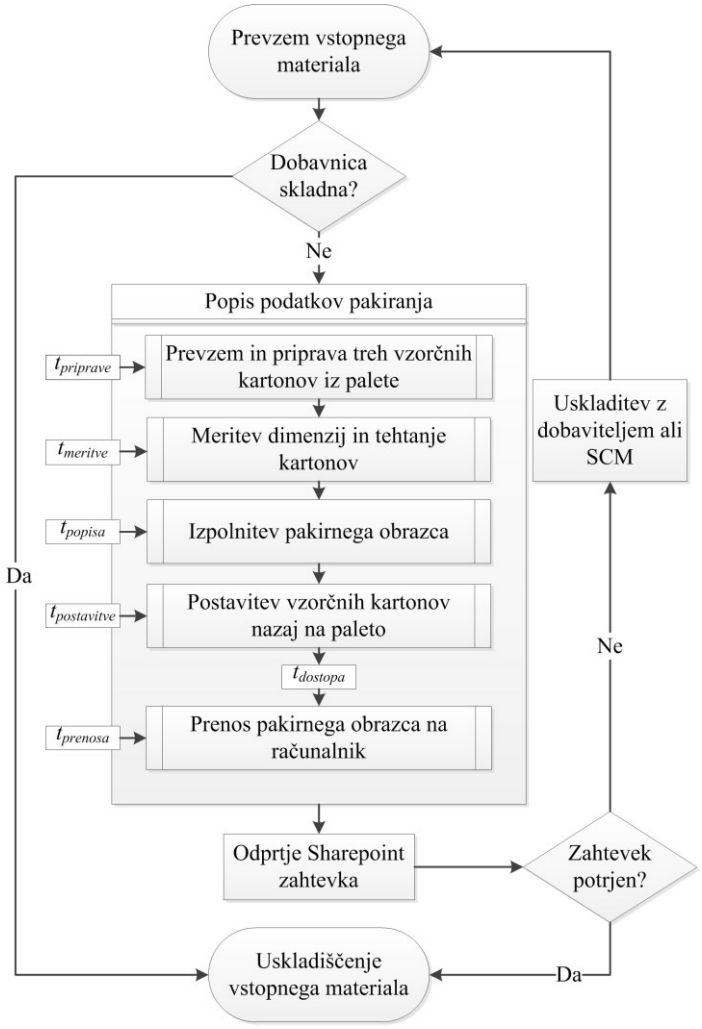
Da bi razvili takšen mobilni sistem, ki bi ga v obstoječe depoje uspešno implementirali in ki bi učinkovito optimiziral izvedbo skladiščnih procesov, moramo biti s slednjimi najprej celovito seznanjeni. V naslednjem poglavju so tako podrobno predstavljeni posamezni koraki trenutnega postopka skladiščenja izdelkov sodelujočega podjetja, ki so tudi nadaljnja osnova za razvoj prototipnega modela.

Poglavje 3 Analiza skladiščnih postopkov podjetja

Skladiščni procesi so bili analizirani v najemnem depoju na lokaciji B iz slike 2.2. Opisani postopki so splošni in veljajo za vsa skladišča sodelujočega podjetja.

3.1 Prevzem blaga

Kot je že opisano v prejšnjem poglavju, je prva skladiščna operacija prevzem blaga. Njene aktivnosti se vrstijo v zaporedju, prikazanem na spodnjem diagramu (slika 3.1).



Slika 3.1: Diagram prevzema vstopnega materiala

Ob dostavi skladiščnik prejme dobavnico v papirnati obliki, na kateri so informacije o vstopnem materialu. Dobavnica je sestavljena iz sledečih podatkov:

- specifikacije zalog (identifikacijska številka šarže ter njena količina),
- naziva in identifikacijske številke končnega izdelka,
- načina pakiranja izdelka (ta podatek je lahko tudi manjkajoč),
- pogojev shranjevanja (temperatura, vlažnost itd.),
- ter drugih podatkov, ki so del naročila.

Skladiščnik mora pred raztovarjanjem in uskladiščenjem blaga preveriti, ali je prejeta dobavnica skladna z naročilom v sistemu ter tako imenovanimi pakirnimi navodili. Pakirna navodila so podatkovni paket, shranjen v lokalni podatkovni bazi, ki vsebuje podrobne informacije o pakiranju izdelka, in sicer:

- količino vsebovanih izdelkov znotraj embalaže,
- identifikacijsko številko in dimenzijo kosa izdelka,
- identifikacijsko številko, dimenzijo in težo enega polnega kartona,
- število kartonov na paleti,
- število slojev kartonov na paleti,
- identifikacijsko številko, dimenzijo in polno težo palete.

Služijo za že prej omenjen sistemski nadzor oskrbe nad skladiščenjem, njihova uporabnost pa je podrobneje opisana v fazi odpreme. Če so podatki v podatkovni bazi skladni s prejetim blagom, se ta lahko prevzame in uskladišči, v nasprotnem primeru je potreben popis podatkov pakiranja.

3.1.1 Postopek popisa podatkov pakiranja

V kolikor dobavnica že vsebuje vse potrebne podatke o načinu pakiranja vstopnega materiala, se ob popisu uporabijo le te, drugače je obvezna ročna preverba in vnos. Pri slednjem scenariju mora skladiščnik najprej razpreti eno izmed prejetih palet ter vzeti tri embalažne elemente za vzorec, kot je predstavljeno na diagramu prevzema (slika 3.1). Te odnese v kontrolni prostor, kjer ima potrebna orodja za preverbo. V primeru, da gre za zunanji izdelek, je poleg tehtanja

embalažnih elementov potrebno še merjenje dimenzij kartona (dolžine, širine in višine), v nasprotnem primeru pa si samo zapiše identifikacijsko številko transportne embalaže, ki je na njej natisnjena. Vsi interni proizvodi uporabljajo namreč embalažo, ki ima že vzdrževano lastno specifikacijo v lokalnem sistemu z vsemi potrebnimi podatki. Prav tako preveri količino elementov v embalaži, ki jo razbere iz etikete, prilepljene na kartonu, ter način zlaganja na paleto (vizualno preveri število kartonov na paleti ter število slojev kartonov na paleti). Izjemoma se pri zunanjih izdelkih lahko zgodi, da so etikete na embalažnih elementih pomanjkljivo vzdrževane ali pa celo ne obstajajo. V takšnih primerih je potrebno razpreti še embalažni element ter ročno preveriti njegovo vsebino.

V skladu s splošnimi postopki podjetja mora skladiščnik zbrane podatke zapisati na listo za popis podatkov pakiranja (slika 3.2).

LISTA ZA POPIS PODATKOV PAKIRANJA

KONČNIH IZDELKOV

ZAHTEVA ZA FIZIČNO PRIPRAVO PODATKOV:

Zahtevo pripravil (ime, priimek): _____

Datum: _____ PODPIS: _____

PODATKI PROIZVAJALCA:

Šifra izdelka: _____ Serija: _____

Naziv:_____

Število prodajnih kosov v originalnem kartonu: _____

PODATKI PREVZEMNIKA:

Šifra transportne embalaže (če gre za lokalno pakiranje): _____

(*zajem podatkov po spodnjem kriteriju, če gre za tujega dobavitelja*):

Dimenzije transportne embalaže:

Dolžina: _____ mm,

Širina: _____ mm,

Višina: _____ mm,

BRUTO teža transportnega kartona

(vedno tehtaj tri naključno izbrane kartone):

TEŽA 1:_____ g,

TEŽA 2: _____ g,

TEŽA 3: _____ g,

Povprečna bruto teža treh embalažnih enot: _____ g,

Število kartonov na polni paleti: _____

Število slojev kartonov na paleti: _____

Datum: Podpis izvajalca:

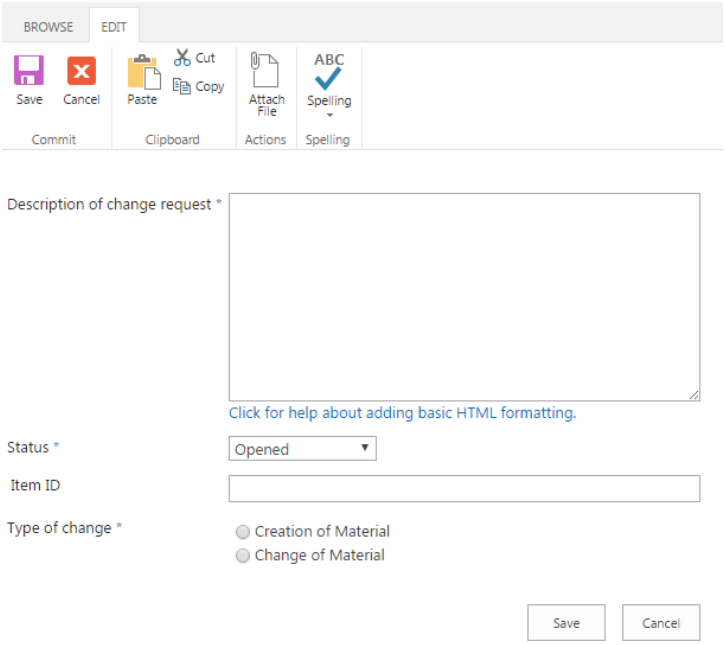
Slika 3.2: Lista za popis podatkov pakiranja

Poleg vseh pakirnih podatkov je za popis obrazca potreben še podatek o izvajalcu zahtevka, ki z lastnim podpisom jamči in nosi odgovornost za točnost posredovanih podatkov. V celoti izpolnjeno listo skladiščnik odnese v prevzemno pisarno, kjer se nahaja računalniška oprema. Razporeditev prostorov se od skladišča do skladišča razlikuje, zato je tudi čas dostopa do računalnika ($t_{dostopa}$) variabilen. Nanj mora prenesti kopijo obrazca, original pa shraniti v arhiv. S tem se zaključi postopek popisa pakirnih podatkov.

3.1.2 Odprtje zahtevka na Sharepoint sistemu

Zaradi varnostnih postopkov podjetja in zagotavljanja večje kakovosti podatkov je skladiščnikom omogočen zgolj vpogled v informacijski sistem. Za kakršno koli vzdrževanje v podatkovni bazi skrbijo odgovorni znotraj SCM, zato se jim mora v primeru sprememb posredovati digitalno shranjen pakirni obrazec. Namesto pošiljanja preko elektronske pošte podjetje za komunikacijo uporablja Microsoftovo spletno platformo, imenovano Sharepoint. Gre za oblačno storitev, ki uporabnikom omogoča enostavno ter varno upravljanje dokumentacije preko spletnih brskalnikov [7]. Ena izmed dodanih vrednosti Sharepointa je tudi preprosta postavitev novih spletnih strani, ki si jih lahko oddelki znotraj družbe sami prilagodijo glede na potrebe poslovanja, ne da bi potrebovali programerja.

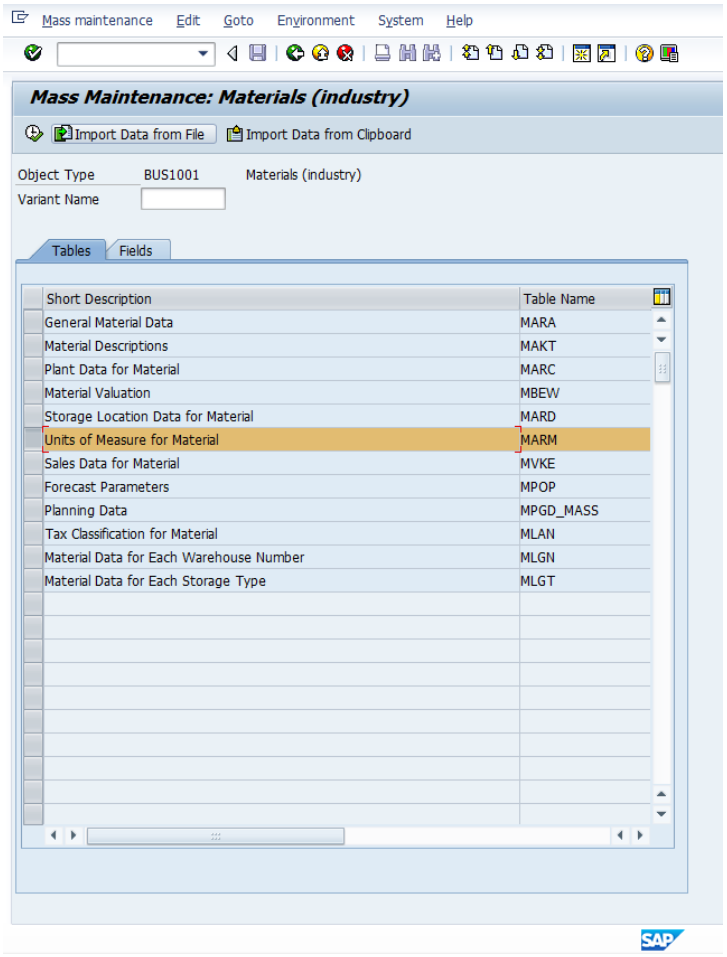
IT oddelek znotraj družbe je za lažje posredovanje pakirnih podatkov znotraj Sharepointa postavil spletno aplikacijo, ki skladiščnikom omogoča ustvarjanje zahtevkov za morebitne popravke v sistemu, kjer lahko pripnejo potrebne dokumente, obenem pa je storitev sinhronizirana s poslovno elektronsko pošto, tako da so ob kreaciji ali zaključku zahtev preko e-mail sporočila obveščene obe strani, tako skladiščniki kot skrbniki podatkov znotraj SCM (slika 3.3).



Slika 3.3: Primer odpiranja Sharepoint zahtevka

3.2 Vzdrževanje pakirnih podatkov znotraj SAP sistema

Znotraj oskrbe obstajata dva oddelka za vzdrževanje osnovnih podatkov. Eden skrbi za podatke na lokalnem nivoju, drugi pa na globalnem. Oba sta zadolžena za zajem, integracijo, ter kasnejšo skupno uporabo točnih, pravočasnih, skladnih in celovitih osnovnih podatkov podjetja, med njimi tudi pakirnih [8]. Družba ima podatkovno bazo vzpostavljeno na svetovno znanem ERP sistemu SAP, razvito v nemškem podjetju SAP SE (slika 3.4). Velja za vodilnega v svetu poslovnih aplikacij kar zadeva programsko opremo in z njo povezanih storitvenih prihodkov. Na podlagi tržne kapitalizacije, velja za tretjega največjega neodvisnega proizvajalca programske opreme na svetu. [9]

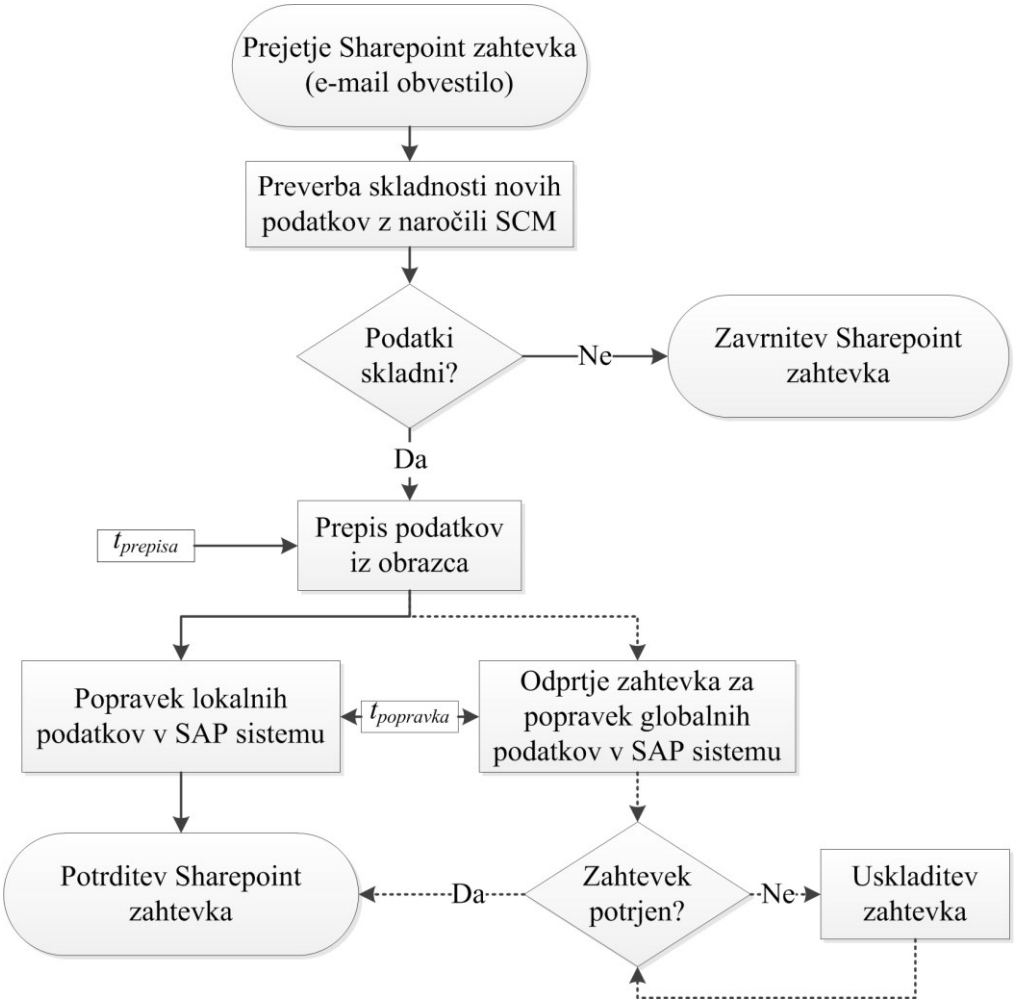


Slika 3.4: Primer uporabniškega vmesnika SAP ERP sistema

Skrbniki osnovnih podatkov na lokalnem nivoju prejmejo obvestilo o novo odprtem Sharepoint zahtevku s strani skladiščnikov. Najprej preverijo skladnost pridobljenih pakirnih podatkov z naročili oskrbe v sistemu. Če so neustrezni, kar je sicer redko, se zahtevek zavrne in je potrebna dodatna uskladitev z drugimi oddelki znotraj SCM ter skladiščem. V kolikor pa so pravilni, se

aktivnosti izvršitve zahtevka vrstijo po spodaj prikazanem diagramu (slika 3.5). Prva operacija je prepis pakirnih podatkov iz obrazca v digitalno obliko. Prej omenjena pakirna navodila se lahko vzdržujejo na lokalnem nivoju, medtem ko se globalni podatki, kot je na primer osnovna teža izdelka, vzdržujejo s strani oddelka, ki skrbi za podatke na globalnem nivoju. Kadar je potrebno vzdrževati tudi globalne podatke, steče večstopenjsko potrjevanje skladiščnih zahtevkov. Oddelek za vzdrževanje podatkov na lokalnem nivoju mora v tem primeru odpreti poseben globalni zahtevek znotraj SAP-a. Šele ko ga oddelek za vzdrževanje podatkov na globalnem nivoju potrdi, se lahko zaključi postopek vzdrževanja, zato se čas popravka od primera do primera razlikuje ($t_{popravka}$).

Ko je posodobitev podatkovne baze končana, skrbniki osnovnih podatkov potrdijo Sharepoint zahtevek, skladiščniki pa s tem dobijo zeleno luč za namestitev prevzetega blaga v skladišče.



Slika 3.5: Diagram poteka aktivnosti skrbnikov osnovnih podatkov

3.3 Uskladiščenje vstopnega materiala

Preden se prejeto blago namesti v skladišče, je potrebno vsako paleto ustrezno označiti z nalepkami ter transportnim nalogom. Te služijo za uspešno izvedbo nadaljnjih skladiščnih postopkov. Seveda pa to velja zgolj za tuje artikle, lokalni izdelki imajo namreč že nameščeno ustrezno označbo za prepoznavo elementa. Omenjena nalepka vsebuje podatke o samem materialu, medtem ko transportni nalog služi skladiščnikom za informacijo o točnem položaju namestitve blaga znotraj skladišča. Vsak razpoložljiv prostor na regalih je definiran z unikatno identifikacijsko številko, ki je prepoznavna vsem delavcem v depojih.

Nalepke in transportne naloge skladiščnik natisne ob potrjeni skladnosti dobavnice z naročili. Vsaka pripada točno določenem prejetem paletnem elementu. Slednjega identificira na podlagi že obstoječih oznak:

- šifre šarže,
- naziva materiala,
- v polovici primerov pa tudi šifre materiala.

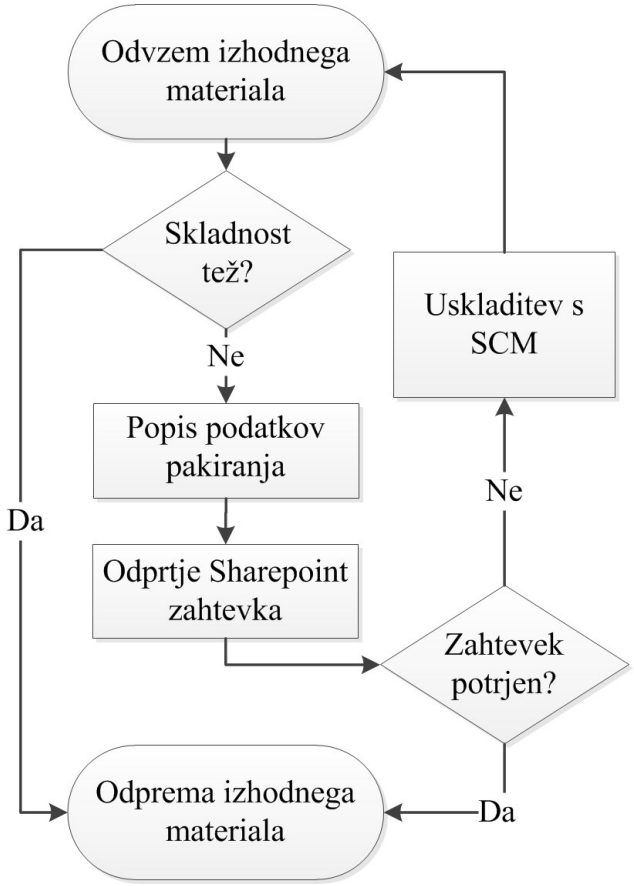
Žal pa te ne zadostujejo za identifikacijo palete s pomočjo čitalcev, zato jih mora skladiščnik ročno preveriti na podlagi prejetega dobavnega obrazca in potrjenega naročila iz sistema. Ko označbe uspešno namesti, viličar prevzame palete in jih uskladišči na ustrezno določena mesta, ki jih določajo transportni nalogi.

3.4 Odprema izhodnega materiala

Pri odpremi materiala ima skladiščnik na viličarju vgrajeno tehtnico in računalnik, ki skupaj delujeta preko interno razvite komponente znotraj sistema SAP. Ta je neposredno povezana z aktualno bazo pakirnih navodil preko internega brezžičnega omrežja, vzpostavljenega znotraj skladišča. Ob odvzemu izhodne palete z regala skladiščnik z optičnim čitalcem odčita črtno kodo na transportnem nalogu. Vzpostavljen sistem na viličarju avtomatsko stehta težo paletnega elementa in jo primerja s trenutno vzdrževano v SAP sistemu. Na LCD zaslonu se izpiše vrednost obeh ter delež odstopanja. Ta je zaradi različnih dejavnikov vedno prisoten. Če je razlika v težah znotraj vnaprej določene dopustne mere, se izdelek lahko odpremi, drugače pa je potreben popis podatkov pakiranja, ki je identičen popisu pri postopku prevzema (slika 3.6).

Do tega lahko pride zaradi več razlogov:

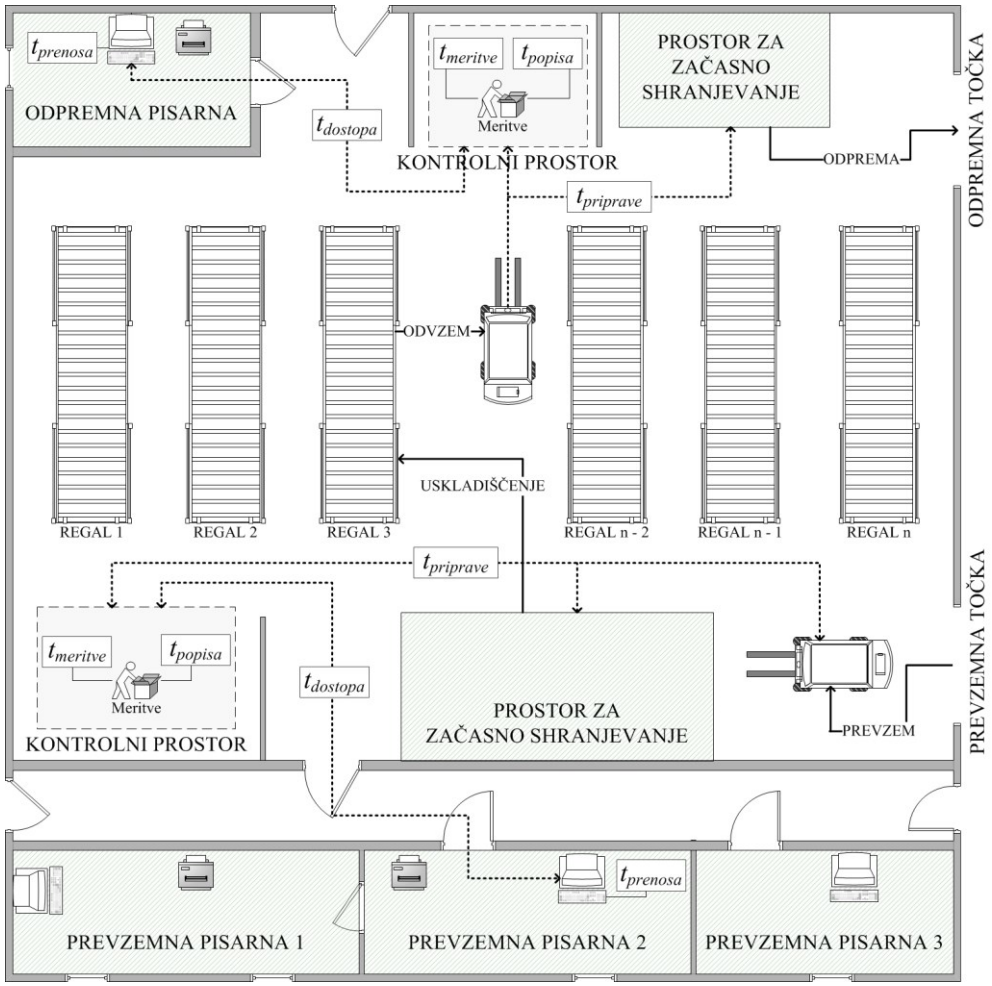
- če skladiščniki pri prevzemu spregledajo spremembo pakiranja izdelka,
- če je količina izdelkov pravilno vzdrževana, a je zaradi uporabe drugačne embalaže teža bistveno spremenjena, kar se pri prevzemu ne opazi,
- pri navzkrižnem pretovarjanju, kjer ni predhodnih preverb ter
- v izjemnih primerih, če je izdelek poškodovan, je znotraj kartona drugače pakiran kot je navedeno, ali če pride do spremembe v sami teži transportne embalaže zaradi vpliva zunanjega okolja (npr. vlage).



Slika 3.6: Diagram odpreme izhodnega materiala

3.5 Kritične točke skladiščenja ter možnosti izboljšav

Za lažjo predstavo celotnega poteka skladiščenja je na sliki 3.7 simbolično prikazana prostorska shema ter izvajajoče operacije znotraj najemnega depoja 1 na lokaciji B. Tu so izvajalci tudi najbolj aktivno sodelovali pri nadaljnji raziskavi.



Slika 3.7: Prostorska shema skladišča na lokaciji B

Predstavljajmo si, da obstajata dve skupini skladiščnih procesov. V prvo spadajo tisti, ki se morajo brezpogojno izvesti pri vsakem vstopajočem materialu. To je začetni prevzem, ki vključuje preverbo skladnosti dobavnic z naročilom, potem uskladiščenje izdelka, če izvzamemo navzkrižno pretovarjanje, ter odvzem in odprema blaga. Ti dogodki se vedno zgodijo in bi se jih dalo optimizirati zgolj z večjo investicijo v avtomatizacijo posameznih procedur, kar pa je za obstoječo podjetno politiko, kot je opisano že prej, neizvedljivo. Zato bomo dali poudarek na optimizaciji postopkov znotraj druge skupine, h kateri štejemo tiste operacije, ki ob morebitnih odstopih povzročajo dodatne časovne zakasnitve, te pa so:

- popis podatkov pakiranja ob prevzemu,
- tiskanje nalepk in transportnih nalogov pri prevzemu ter
- popis podatkov pakiranja pri odpremi.

Slednja je še posebej kritična, saj se ob daljših zakasnitvah lahko čas odpreme zamakne za cel dan, kar prinaša podjetju dodatne stroške ter na koncu nezadovoljstvo strank, če blago ni dostavljeno v predvidenem roku. Posledično lahko to negativno vpliva na nadaljnje poslovanje s končnim kupcem in s tem postavlja družbo v neugoden položaj.

Največji problem pri sodelujočih skladiščih predstavlja dostopnost računalniške odpreme. Ob vsakem odstopu mora skladiščnik najprej opraviti meritve v kontrolnem prostoru, nato pa zapisane pakirne podatke odnesti do pisarne. Do nje pa povprečen čas dostopa ($t_{dostopa}$) lahko traja tudi po več minut, če se nahaja na čisto drugem koncu kot prevzemna ali odpremna točka. Poleg tega pa sta skeniranje obrazca in prenos na sistem Sharepoint spet zamudna ($t_{prenosa}$), saj ima slednji z vidika uporabnikov velikokrat slabo odzivnost. Idealna rešitev bi na tem mestu bila uporaba pametnih telefonov, saj jih ima večina skladiščnikov nenehno ob sebi, so prenosljivi, lahko se povežejo na obstoječe brezžično omrežje v skladišču, z vgrajeno tehnologijo pa ponujajo vrsto različnih možnosti za učinkovito posredovanje potrebnih informacij.

Pametni telefoni bi lahko z ustrezno mobilno aplikacijo, ki bi omogočala preverjanje pristnosti uporabnika, odpravili potrebo po obveznem papirnatem izpolnjevanju pakirnih obrazcev, kar bi bilo ugodno tudi za skrbnike osnovnih podatkov, ki morajo ročno prepisati podatke v digitalno obliko, preden jo naložijo v SAP-ovo podatkovno bazo ($t_{prepisa}$). Dodano vrednost bi lahko prispevala tudi ustrezna oblika zapisa pakirnih podatkov v tip datoteke, ki bi se lahko neposredno prenesla v SAP brez predhodne priprave.

Tiskanje nalepk in transportnih nalogov pa zaenkrat ostaja odprta tema, saj na zunanjih artiklih ni ustreznega identifikatorja, s katerim bi lahko sistem na viličarju uspešno zaznal izdelek in preveril skladnost tež. Rešitev bi lahko bila v OCR tehnologiji, ki pa zaradi kompleksnosti implementacije v obstoječe skladiščno okolje z diplomskim delom ni bila raziskana.

Poglavje 4 Obstoječe mobilne skladiščne rešitve

Na spletu lahko najdemo vrsto obstoječih rešitev za optimizacijo skladiščnih procesov s pomočjo mobilne tehnologije, kar je glede na njen nenehni se razvoj ter povečanje dostopnosti in uporabnosti pri poslovanju podjetij povsem pričakovano. Še vedno pa je spletni trg zaradi svoje odprtosti in visoke konkurenčnosti nasičen s kopico neuporabnih izdelkov, kar družbam daje nezaupanje ter postavlja dvome o smiselnosti implementacije mobilnih produktov.

Glede na podano analizo obravnavanega skladiščnega okolja mora mobilna platforma za uspešno odpravo pereče skladiščne problematike omogočati naslednje osnovne funkcionalnosti:

1. vzpostavljen sistem na zunanjem strežniku, ki omogoča centralni dostop in upravljanje podatkov s strani različnih skladišč ter oddelkov podjetja,
2. preverjanje pristnosti uporabnikov,
3. možnost odpiranja, spreminjanja in zaključevanja skladiščnih zahtevkov,
4. možnost shranjevanja ter kasnejšega vpogleda zahtevkov,
5. prilagodljivost sestave zahtevka znotraj aplikacije,
6. dodeljevanje vlog posameznemu uporabniku glede na delovno mesto (izvajalec ali potrjevalec zahtevkov),
7. varnost podatkov,
8. enostavnost namestitve in uporabe,
9. finančno ugodna rešitev,
10. vgrajen sistem pošiljanja obvestil ob spremembah ali ustvarjanju zahtevkov ustrezni skupini uporabnikov (izvajalcem ali potrjevalcem zahtevkov),
11. omogočeno delovanje ob izpadu omrežja (brez povezave),

12. možnost prenosa pakirnih podatkov v ustrezno obliko zapisa, primerno za interno podatkovno bazo.

V nadaljevanju so predstavljeni obstoječi mobilni sistemi treh različnih ponudnikov. Na podlagi kriterijev, opredeljenih v prejšnjem odstavku, smo jih poiskali na spletu. Izmed vseh preučenih obstoječih skladiščnih rešitev so se omenjeni trije najbolj približali opredeljenim kriterijem, na podlagi katerih smo podali tudi ocenjeno primernost teh rešitev za sodelujoče podjetje. Ključne razlike med njimi so v določenih funkcionalnostih, ki jih ponujajo. Preostalih rešitev nismo opisovali, saj so bile po karakteristikah manj ustrezne od predstavljenih.

4.1 WarehouseOS

Gre za oblačno storitev, ki ponuja izviren način upravljanja skladiščnih operacij s pomočjo mobilnih naprav. Med drugim omogoča komunikacijo z obstoječimi ERP sistemi, kar bi lahko rešilo problem prenosa celovitih pakirnih podatkov v ustrezni format, hkrati pa zagotavlja nizke stroške integracije. Naprave delujejo skupaj z RFID ali optičnimi čitalci, uporabniški vmesniki pa so pregledni in s tem dajejo enostavnost uporabe aplikacije. [10]

Žal pa WarehouseOS ne ponuja možnosti upravljanja s skladiščnimi zahtevki s strani različnih oddelkov podjetja, temveč služi bolj za kontrolo in obvladovanje zalog uskladiščenega blaga s samostojnim, neodvisnim skladiščnim sistemom, kar pa ne bi rešilo zgoraj omenjene problematike.

4.2 Snappii

Podjetje Snapii je ustvarilo platformo za razvijanje funkcionalno bogatih poslovno-mobilnih aplikacij brez pisanja kakršne koli kode. Tako programerjem kot ne-programerjem omogoča relativno hitro postavitev sistema brez predhodnega znanja programiranja. S pomočjo uporabniku prijaznega spletnega urejevalnika se lahko mobilno aplikacijo oblikuje in objavi zgolj v nekaj minutah. Za skladiščne procese ima že vnaprej pripravljene predloge, ki se lahko uporabijo in prilagodijo. Je tudi ena izmed redkih, ki ponuja brezplačno testiranje. [11]

Ob preizkusu njenega delovanja smo prišli do sledečih ugotovitev. Aplikacija je enostavna za namestitev in uporabo, omogoča izpolnjevanje pakirnih obrazcev, prilagoditev njihove vsebine uporabniškim zahtevam ter ima implementirane skoraj vse zgoraj omenjene funkcionalnosti. Problem pa je v njeni namenskosti, namreč cilj Snappii-ja je, podobno kot pri WarehouseOS, v beleženju in pregledu zalog posameznih izdelkov v depojih, ne pa v ustvarjanju in potrjevanju skladiščnih zahtev za posodabljanje interne podatkovne baze. Poleg tega ne omogoča prenosa

pakirnih podatkov v ustrezno obliko zapisa, temveč ga shrani neposredno v PDF obliko. Zato za dejansko uporabo ne bi bil primeren.

4.3 RFgen

Izmed nabora obstoječih je bilo smiselno analizirati še programsko opremo RFgen, ki je za razliko od ostalih integrirana s SAP sistemom. Uporabnikom ponuja lasten razvojni programski studio, imenovan »RFgen mobile development studio«, ki omogoča internim razvijalcem samostojno kreiranje mobilnih aplikacij ter njihovo prilagajanje glede na potrebe družbe. RFgen bi lahko bil učinkovita rešitev, v kolikor bi se znotraj SAP-a razvila transakcija, ki bi omogočala večfazno potrjevanje sprememb pakiranja izdelka. Tako bi ob odobritvi zahtev skladiščnikov s strani skrbnikov osnovnih podatkov stekla avtomatska posodobitev pakirnih podatkov v sistemu. RFgen pa bi poskrbel za prilagoditev uporabe razvite transakcije na mobilnih napravah. [12]

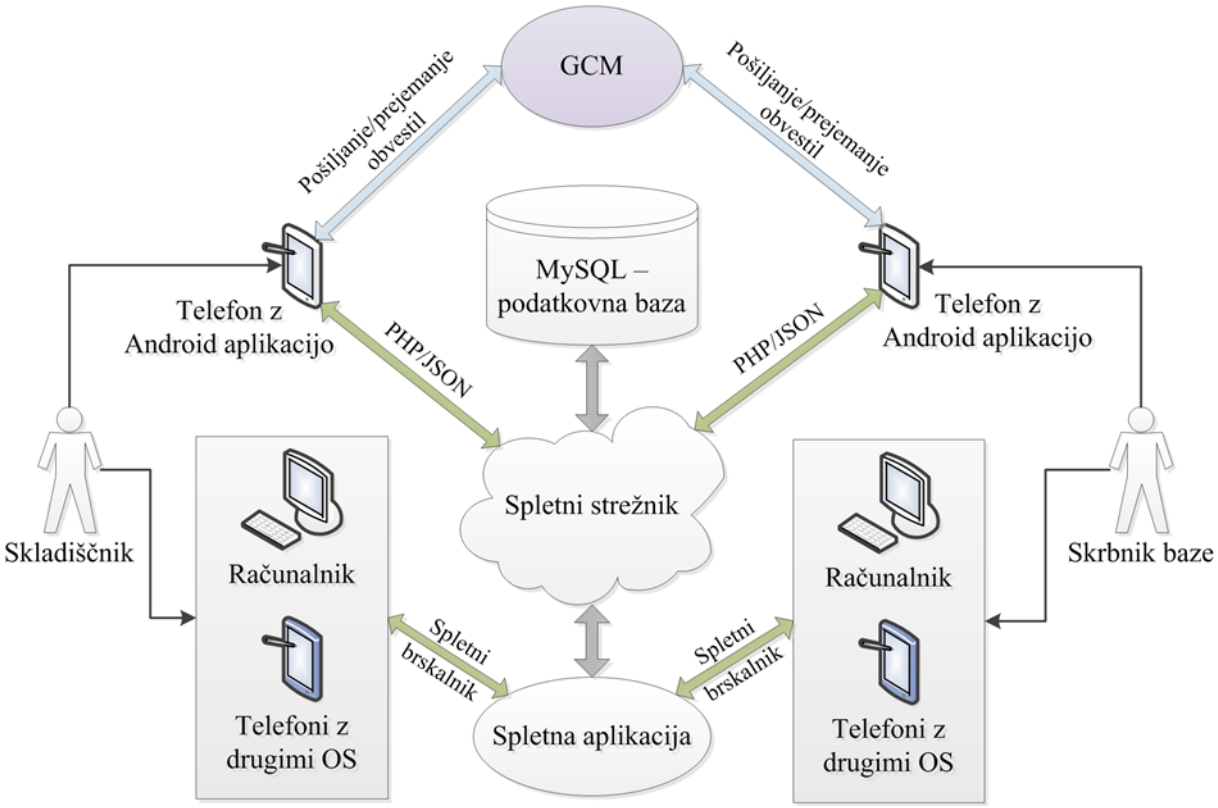
Seveda pa je takšna namestitev izredno kompleksna, zahteva veliko strokovnega znanja in je posledično zelo draga za relativno enostaven problem, kar pa bi bilo za sodelujoče podjetje nesprejemljivo.

4.4 Povzetek uporabnosti analiziranih obstoječih sistemov

Glede na raziskane obstoječe skladiščne mobilne aplikacije lahko povzamemo, da se je večina kljub širokem naboru ponudb na trgu usmerila v razvoj neodvisnih samostojnih sistemov, ki ponujajo modernejše, mobilne načine upravljanja s skladiščnimi zalogami, kar, globalno gledano, lahko dobro optimizira samo izvedbo skladiščnih postopkov. Žal pa nobena ne ponuja učinkovitega načina posredovanja celovitih pakirnih podatkov odgovornim osebam za vzdrževanje in posodabljanje že obstoječe podatkovne baze ali pa je njihova implementacija enostavno predraga ter preveč kompleksna. Zato je bil v okviru diplomskega dela razvit nov prototipni mobilni sistem, prilagojen glede na obstoječe skladiščno okolje in z vgrajenim naborom osnovnih funkcionalnosti, potrebnih za dejansko testiranje.

Poglavje 5 Razvoj prototipnega informacijskega sistema

V tem poglavju so predstavljeni posamezni koraki razvoja prototipnega informacijskega sistema za optimizacijo prej analiziranih skladiščnih postopkov ter opis namestitve njegovih osnovnih komponent. Celovit vpogled delovanja razvite platforme je prikazan na sliki 5.1, ki je hkrati načrtna osnova postavitve sistema.



Slika 5.1: Shema postavljenega prototipnega sistema

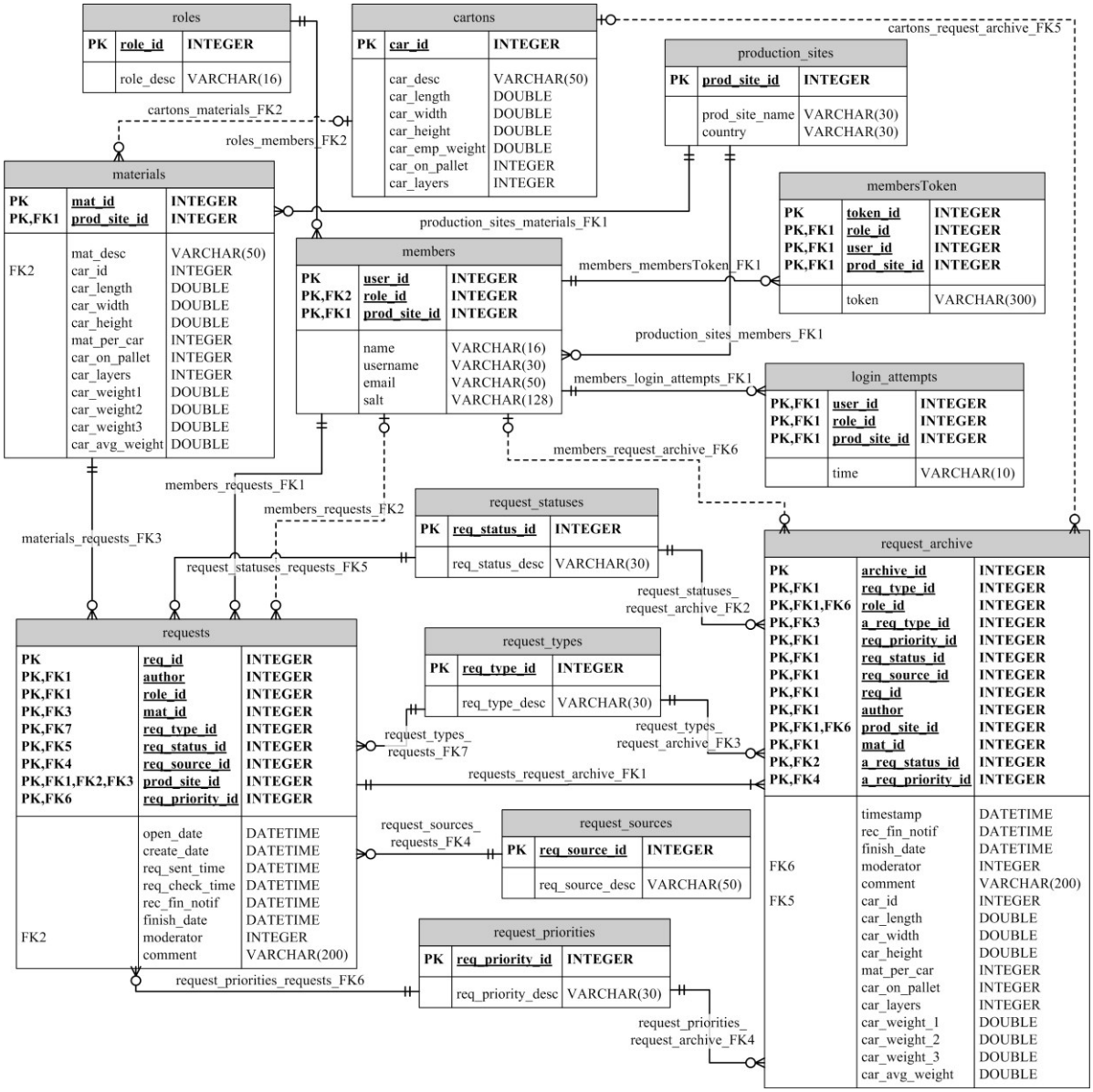
5.1 Spletni strežnik

Testno programsko okolje je bilo postavljeno na zunanjem spletnem strežniku, imenovanem »000webhost.com« [13]. Njegova platforma omogoča storitev brezplačnega gostovanja z možnostjo razvoja MySQL podatkovnih baz ter spletnih strani na pod-domenah, internetna komunikacija z zunanjimi napravami pa poteka preko PHP skript. Prav tako ima vgrajeno

uporabniku prijazno administratorsko konzolo, ki omogoča hitro, enostavno namestitev spletnih aplikacij ter njihovih funkcij, kar je zadostovalo potrebam za izdelavo prototipa.

5.2 Zgradba podatkovnega modela

Podatkovni model je bil zasnovan na podlagi liste za popis pakirnih podatkov iz slike 3.2 ter obstoječe postavljene spletne strani na sistemu Sharepoint (slika 3.3). Tako so nastale tri osnovne podatkovne entitete: izdelki (*materials*), zahtevki (*requests*) in embalažni elementi (*cartons*). Ker je bilo potrebno implementirati tudi prijavo v sistem ter vloge uporabnika, je bila definirana še četrta osnovna entiteta, uporabniki (*members*). Celotna struktura podatkovnega modela z vsemi potrebnimi relacijami in tabelami je prikazana na spodnji sliki 5.2, podrobni opisi posameznih entitet pa so predstavljeni v tabeli 5.1.



Slika 5.2: Shema podatkovnega modela

Naziv tabele	**Bazni naziv**	**Vsebina tabele ter njen namen**
Uporabniki	*members*	Podatki o uporabniku sistema za preverjanje pristnosti ob prijavi ter njegovo identifikacijo
Izdelki	*materials*	Informacija o trenutnem načinu pakiranja izdelka ter identifikacijski podatki materiala
Zahtevki	*requests*	Specifikacije zahtevka ter časovne spremenljivke oz. žigi, definirani ob določenih dogodkih izvajanja zahtevka, uporabljeni tudi za analizo testiranja
Embalažni elementi	*cartons*	Podatki o obstoječih embalažnih elementih iz lokalne podatkovne baze, ki omogočajo predhodno pridobitev določenih pakirnih količin
Arhiv sprememb zahtevkov	*request_archive*	Beležena zgodovina sprememb zahtevkov in materialov, zaradi poenostavitve združena v eno tabelo
Ključi uporabnika	*membersToken*	Podatek o ključu uporabnika, ki služi za ustrezno identifikacijo naprave uporabnika ob pošiljanju obvestil
Stanje zahtevka	*request_statuses*	Definira, ali je zahtevek v odprtem, med-procesnem ali zaključenem stanju (*open, in-process, feedback needed, cancelled, finished*)
Tip zahtevka	*request_types*	Določa, ali gre za vnos novih pakirnih podatkov v sistem ali za popravek obstoječih (*creation, change*)
Izvor zahtevka	*request_sources*	Informacija, potrebna za testiranja, ki določa, ali je vnos zahtevka potekal preko mobilne ali spletne aplikacije
Prioriteta zahtevka	*request_priorities*	Podatek o nujnosti izvedbe zahtevka, prednostno prevzemanje ali po sistemu FIFO (*regular, high, critical*)
Obrati proizvodnje	*production_sites*	Vsebuje informacijo o izvoru izdelka ali uporabnika (za potrebe testiranja uporabljena zgolj za identifikacijo skladiščne lokacije)
Poskusi prijav	*login_attempts*	Informacija o številu neuspešnih prijav uporabnika ob morebitnem poskusu vdora v fazi testiranja
Uporabniške vloge	*roles*	Definira uporabnikovo vlogo za dodeljevanje sistemskih pravic

Tabela 5.1: Vsebina podatkovnega modela

V nadaljevanju so izpostavljene ključne točke pri strukturiranju podatkovne baze.

5.2.1 Uporabniške vloge

Umestitev ustreznih uporabniških vlog (*roles*) je bila zelo pomembna za razvoj prototipa, saj so predstavljale podlago za dodelitev sistemskih pravic in dovoljenj za izvršitev posameznih operacij znotraj aplikacije.

Za testno okolje so bile ustvarjene tri različne vloge:

1. Vloga skladiščnika (»*whworker*«): lahko ustvari zahtevek, popravlja lastne odprte zahtevke, vendar jih ne more sam zaključiti, razen v primeru preklica, ter ima možnost vpogleda vseh ustvarjenih zahtevkov, tako svojih kot od ostalih uporabnikov sistema.

2. Vloga skrbnika osnovnih podatkov (»*manager*«): lahko ustvari, pregleduje, popravlja in zaključuje vse odprte zahtevke; ko je enkrat izvedba zahtevka končana, je možen samo še vpogled.

3. Vloga administratorja (»*admin*«): poleg vseh pravic, ki jih ima skrbnik osnovnih podatkov, lahko ustvari nove uporabniške račune ter popravlja že zaključene zahtevke.

5.2.2 Definicija časovnih spremenljivk zahtevka

Za rezultate in analizo testiranja so bili ključni podatki specifične časovne spremenljivke oziroma žigi. Vsi so bili nastavljeni na »DATETIME« format, ki združuje tako datum kot uro, definirani pa znotraj tabele zahtevkov:

1. Čas pričetka operacije odpiranja zahtevka (*open_date*): čas, ko je uporabnik pričel z aktivnostjo ustvarjanja zahtevka, a ga še ni izpolnil in potrdil.

2. Čas zaključka operacije odpiranja zahtevka (*create_date*): čas, ko je uporabnik končal z aktivnostjo ustvarjanja zahtevka, zahtevek je izpolnjen in potrjen.

3. Čas uspešno poslanega novo odprtega zahtevka (*req_sent_time*): čas, ko je bil zahtevek uspešno poslan in shranjen v podatkovno bazo.

4. Čas prejetega obvestila o novo odprtem zahtevku (*req_check_time*): čas, ko so skrbniki osnovnih podatkov prejeli obvestilo o novem skladiščnem zahtevku na napravo.

5. Čas potrditve zaključka zahtevka (*finish_date*): čas, ko je skrbnik osnovnih podatkov uspešno zaključil zahtevek.

6. Čas prejetega obvestila o zaključenem zahtevku (*rec_fin_notif*): čas, ko je skladiščnik prejel obvestilo o njegovem zaključenem zahtevku.

Zgoraj definirani časi služijo za pridobitev sledečih pomembnih analitičnih informacij:

- $t_{izpolnitve}$: čas za izpolnitev zahtevka (pri analizi je primerjan s t_{popisa} iz slike 3.1). Koliko časa potrebujejo skladiščniki za izpolnitev zahtevka, nam pove časovna razlika med prvima dvema časoma (*create_date – open_date*).

- $t_{posredovanja}$: čas, potreben za posredovanje zahtevka na strežnik. Je načeloma zanemarljiv, razen če pride do izpada omrežja in zahtevka ni bilo mogoče ob potrditvi kreacije istočasno uspešno poslati. Pridobi se ga z razliko druge in tretje časovne spremenljivke (*req_sent_time - create_date*).

- $t_{obvestila1}$: čas, potreben za uspešno dostavo obvestila skrbnikom osnovnih podatkov o novo odprtem zahtevku. Določa ga časovni razpon med tretjim in četrtim časovnim žigom (*req_check_time - req_sent_time*).

- $t_{obvestila2}$: čas, potreben za uspešno dostavo obvestila skladiščniku o zaključku njegovega zahtevka. Definira ga razlika med petim in šestim časom (*rec_fin_notif – finish_date*).

Njihova dejanska uporaba pa je prestavljena v naslednjem poglavju.

5.2.3 Stanja zahtevkov

Stanje zahtevka je kritičen podatek, saj definira, v kateri fazi je izvedba zahtevka in kako le-ta napreduje, kar je v primeru urgenc ključna informacija za skladiščnike. Delimo jih na dve večji skupini:

1. Odprto stanje: Zahtevek je odprt, kadar je na novo ustvarjen (*open*), v fazi obdelave (*in process*) ali zavrnjenem stanju (*feedback needed*), v kolikor so potrebne dodatne informacije oz. usklajevanja s strani skladiščnikov.

2. Zaključeno stanje: Zahtevek je zaključen, ko je popravek uspešno izveden (*finished*) ali pa se sprememba podatkov ne odobri oz. ni potrebna (*cancelled*).

Status lahko spreminja zgolj izvedenec zahtevka, torej skrbnik osnovnih podatkov. Izjemoma, če skladiščnik po potrditvi ugotovi, da je zahteva nepotrebna ali napačna, lahko odprt zahtevek prekliče.

5.2.4 Arhiviranje sprememb zahtevkov

Ker se v fazi izvršitve zahtevka spremeni minimalno vsaj njegovo zgoraj opisano stanje, v primeru napak vnosa pa tudi vsebovani pakirni podatki, je bilo potrebno definirati entiteto (*request_archive*), ki beleži vsako narejeno spremembo na zahtevku ali materialu, ter čas njene izvedbe (polje *timestamp*). Arhiv je zaradi poenostavitve, ki zadostuje testnemu okolju, sestavljen iz združene tabele izdelkov in zahtevkov, izpuščena so bila zgolj fiksna polja obeh entitet, ki se v fazi obdelave skladiščnih zahtev ne morejo spreminjati.

Beležen čas izvedbe sprememb (*timestamp*) bi potencialno lahko služil za izračun še ene analitične časovne spremenljivke, in sicer $t_{popravka}$ (iz slike 3.5). Kot je bilo že prej opisano, je namreč čas obdelave zahtevka s strani skrbnikov osnovnih podatkov odvisen predvsem od tega, ali gre zgolj za popravek lokalnih ali tudi globalnih podatkov. Za fazo testiranja je bil sklenjen dogovor, da se v primeru zaključka obveznosti s strani skrbnikov lokalnih osnovnih podatkov in čakanja na odobritev s strani oddelka za vzdrževanje globalnih podatkov vsak zahtevek najprej spremeni v stanje obdelave (*in process*) in šele ob globalni potrditvi v zaključen status (*finished*). Tako se je zabeležil ustrezni časovni žig, časovna razlika med njim in časom prejetega obvestila o novem odprtem zahtevku (*req_check_time – timestamp*) pa je predstavljala zgoraj omenjen čas popravka. Ker so statusi zahtevka definirani enako kot na Sharepoint strani, je za potrebe analize postopek izvedbe Sharepoint zahtevkov enak.

5.2.5 Beleženje pakirnih podatkov

Za beleženje pakirnih podatkov sta se uporabili entiteti izdelkov (*materials*) ter embalažnih elementov (*cartons*). Kot lahko opazimo iz slike 5.2, je vsebina obeh zelo podobna. Razlog za to je, da tabela kartonov beleži pakirne podatke obstoječih embalažnih elementov, uporabljenih znotraj družbe. Ti imajo že vnaprej določeno svojo identifikacijsko številko, kot je že bilo opisano v tretjem poglavju. V primeru, da gre za lokalni izdelek, se določeni podatki embalažnega elementa lahko uporabijo za pakirne podatke materiala neposredno iz tabele kartonov. Če pa gre za zunanje blago, jih je potrebno definirati v sklopu podatkov izdelka, saj so zunanje embalaže preveč raznolike in je shranjevanje specifik posamičnih kartonov v ločeno tabelo kartonov nesmiselno.

5.2.6 Postavitev podatkovnega modela na strežnik

Zgrajen podatkovni model je bil za testiranje postavljen na odprtokodni podatkovni bazi MySQL uporabljenega strežnika. Njeno upravljanje je potekalo preko spletnega orodja phpMyAdmin, dostopnega na strežnikovi konzoli. Ker se je uporabila brezplačna licenca spletnega gostovanja, je ta omejena zgolj na gradnjo ne-relacijskih podatkovnih modelov, kar

sicer ni bila ovira za razvoj in testiranje prototipa, se pa od predstavljene relacijske normalizirane podatkovne baze (slika 5.2) v določenih segmentih razlikuje. Tako povezave s primarnimi in tujimi ključi med podatkovnimi tabelami niso obstajale, sestavni deli skladiščnikovih zahtevkov, kot so stanje (*request_statuses*), prioriteta (*request_priorities*), izvor (*request_sources*) ter tip (*request_types*), pa so bili zaradi lažjih poizvedb združeni z entiteto zahtevkov (*requests*). Enako je veljalo za uporabniške vloge (*roles*), ki smo jih dodali podatkovni tabeli uporabnikov (*members*). Ker je vsebina entitet ostala povsem enaka, sheme ne-relacijskega podatkovnega modela nismo posebej predstavili.

5.3 Razvoj mobilne aplikacije

Prototipna mobilna aplikacija je bila razvita za mobilne naprave z operacijskim sistemom Android [14]. Slednji je za razliko od drugih OS odprtokoden, ponuja brezplačno uporabo lastne razvojne programske opreme Android Studio in je tudi daleč najbolj razširjen med uporabniki pametnih telefonov. Njegov svetovni delež v letu 2016 znaša kar 84,1 odstotkov, sledi mu iOS s 14,8 odstotki, medtem ko je imetnikov Windows telefonov manj kot odstotek. [15]

Po uspešni vzpostavitvi z zunanjim strežnikom in zgradbi ustreznega podatkovnega modela je glede na zahtevane osnovne funkcionalnosti sistema, podane v četrtem poglavju, sledila izdelava aplikacije za pametne telefone. Ta omogoča odpiranje in posredovanje skladiščnih zahtevkov neposredno iz kontrolnega prostora depoja, kjer potekajo meritve in popisi aktualnih pakirnih podatkov izdelkov. V nadaljevanju so predstavljene njene ključne implementacije za uspešno izvedbo vgrajenih aktivnosti.

5.3.1 Dostopanje do strežnika

Dostop do strežnika je bil eden izmed osnovnih pogojev za delovanje aplikacije. Povezava je potekala preko standardnega Java razreda *HttpURLConnection*, ki se uporablja za pošiljanje in prejemanje podatkov preko spleta. Omrežne operacije so včasih dolgotrajne, kar lahko aplikacijam, razvitim na Android platformi, ki je zasnovana na delovanju enovite programske niti uporabniškega vmesnika (*UI thread*), povzroča operativne težave. Zato je bila kakršna koli komunikacija s strežnikom implementirana znotraj razreda z razširjeno funkcijo asinhronih opravil (*AsyncTask*). Ta omogoča izvajanje daljših operacij v ozadju z metodo *doInBackground(Params…)*. Osnovna nit aplikacije lahko tako nemoteno nadaljuje s svojimi tekočimi nalogami. Ko so operacije v ozadju zaključene, se sproži metoda *onPostExecute(Result)*, ki vrne njihov rezultat. [16]

Izsek spodnje kode 5.1 prikazuje primer vzpostavljene povezave s strežnikom z uporabo asinhronih opravil.

```
public class fetchMemberDataAsyncTask extends AsyncTask<Void, Void, Member> {
        //..........
         @Override
         protected Member doInBackground(Void... params){
                Member returnedMember = null;
                //.........
                try {
                //...........
                URL url = new URL(SERVER_ADDRESS + "includes/FetchMemberData.php");
                HttpURLConnection con = (HttpURLConnection) url.openConnection();
                //.........................
                }catch (Exception e) {
                        e.printStackTrace();
                }
                //..........
                return returnedMember;
         }
         @Override
         protected void onPostExecute(Member returnedMember){
            //............
             super.onPostExecute(returnedMember);
         }
}
```

Koda 5.1: Primer uporabe asinhronih opravil

Logika postavljenega sistema komunicira s strežnikom preko PHP skript, ki so na njem naložene. Naloga mobilne aplikacije se je povezati z izbrano skripto preko prej opisanega *HttpUrlConnection* razreda in ji poslati ustrezen podatkovni paket, potreben za SQL poizvedbo, spisano znotraj nje. Za zapis omenjenih paketov se je pri pošiljanju na strežnik uporabil kodiran *HashMap* Java objekt, prilagojen za PHP-POST metodo, medtem ko se je za prejemanje podatkov testiral JSON format, ki je bil ob prejetju pretvorjen v primeren Java objekt s pomočjo knjižnice GSON. Prednost JSON tekstovnega zapisa pred ostalimi je v tem, da je razumljiv tako ljudem kot napravam. Ker je jezikovno neodvisen, je idealen za izmenjavo podatkov. [17] GSON je ena izmed redkih odprtokodnih knjižnic, ki ponuja ustrezno pretvorbo JSON objektov v Java objekte in obratno, prav tako pa je zelo enostavna za uporabo [18]. Primer implementacije prenašanja podatkovnih paketov je predstavljen v kodah 5.2 in 5.3.

```
//... primer pošiljanja podatkov
Map<String,String> dataToSend = new HashMap<>();
dataToSend.put("email", member.email);
dataToSend.put("password", member.password);
String encodedStr = getEncodedData(dataToSend);
//............
HttpURLConnection con = (HttpURLConnection) url.openConnection();
con.setRequestMethod("POST");
con.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(con.getOutputStream());
writer.write(encodedStr);
writer.flush();
//............
//... primer prejemanja podatkov
StringBuilder sb = new StringBuilder();
reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
String result;
while((result = reader.readLine()) != null) {
        sb.append(result);
}
result = sb.toString();
Gson gson = new Gson();
JsonReader jReader = new JsonReader(new StringReader(result));
jReader.setLenient(true);
returnedMember = gson.fromJson(jReader, Member.class);
```

Koda 5.2: Prenos podatkov iz/na strežnik

```
<?php
        //............implementacija POST metode
        if (isset($_POST['email'], $_POST['password'])) {
                //............
        }
        echo json_encode($member); //kodiranje podatka v JSON zapis
?>
```

Koda 5.3: Primer obdelave podatkov v PHP skripti

5.3.2 Preverjanje pristnosti

Za identifikacijo uporabnika sistema je bil vsakemu sodelujočemu dodeljen uporabniški račun z ustrezno določeno vlogo. Njihova vsebina je opisana znotraj entitete uporabnika (*members*) v podatkovnem modelu. Zaradi želje podjetja po anonimnosti so bili za prototip narejeni zgolj testni računi, ki ne razkrivajo identitete zaposlenih. Zato je bilo za testiranje dovolj vzpostaviti osnovni nivo varnega preverjanja pristnosti uporabnikov, namenjenega zgolj za preprečitev morebitnih lažjih vdorov, ki bi potencialno lahko škodovali uspešni izvedbi preizkusa sistema.

Prijava v aplikacijo je tudi njena prva aktivnost. Ob vnosu naslova elektronske pošte in gesla uporabnika, se slednje šifrira s SHA-512 algoritmom. SHA je enosmerna razpršilna iterativna funkcija, ki prevzeto sporočilo obdela v zgoščen kodiran zapis, imenovan »*Message digest*«. Njeni algoritmi zagotavljajo celovitost podatkov in »vsaka sprememba sporočil bo z veliko verjetnostjo povzročila drugačen kodiran zapis.« [19] SHA-512 pa je različica, ki omogoča 64-bitno zapisovanje besed.

Strežnik preveri pristnost uporabnika z implementiranim obstoječim postopkom, ki je podrobneje opisan v [20]. Vsaka uporabniška entiteta vsebuje unikatno, s SHA-512 algoritmom naključno generirano kodo, imenovano *salt*. Ta se s prejetim šifriranim geslom iz aplikacije združi v en niz in ponovno razprši s SHA-512. Za uspešno avtentikacijo mora biti končni rezultat identičen shranjenemu geslu v podatkovni bazi. Del postopka je prikazan v kodi 5.4.

```
<?php
//...........
function login($email, $password, $mysqli) {
    if ($stmt = $mysqli->prepare("SELECT user_id, name, role, username, password, salt,
prod_site_id FROM members WHERE email = ? LIMIT 1")) {
                $stmt->bind_param('s', $email);  // Bind "$email" to parameter.
                $stmt->execute();    // Execute the prepared query.
                $stmt->store_result();
                $stmt->bind_result($user_id, $name, $role, $username, $db_password, $salt,
                $prod_site_id);
                $stmt->fetch();
                $password = hash('sha512', $password . $salt);
                //...........
                if ($db_password == $password) {
                        // Geslo pravilno, avtentikacija uspešna ...........
```

Koda 5.4: Preverba pravilnosti gesla na strežniku

Da uporabnikom ne bi bila potrebna ponovna prijava ob vsakem zagonu aplikacije, se ob prvem vpisu uporabniški podatki shranijo na Androidov vmesnik *SharedPreferences*. Ta ohranja preferenčne informacije tudi po zaustavitvi aplikacije. Podatki se lahko zbrišejo z uporabo metode *clear()* vmesniškega urejevalnika v kodi ali odstranitvijo programa z naprave. [21]

5.3.3 Glavne aktivnosti aplikacije

Po uspešni avtentikaciji se uporabniku prikaže osnovni meni (slika 5.3, levo), ki ponuja izbiro sledečih aktivnosti aplikacije:

1. Odprtje novega zahtevka (*Create request*): aktivnost, namenjena skladiščnikom, za popis podatkov pakiranja (slika 5.3, na sredini).

2. Zahtevki v čakanju (*Pending requests*): aktivnost, ki izpiše seznam neuspešno poslanih zahtevkov zaradi težav s povezljivostjo omrežja.

3. Moji odprti zahtevki (*My open requests*): aktivnost, ki izpiše seznam odprtih zahtevkov uporabnika (slika 5.3, desno).

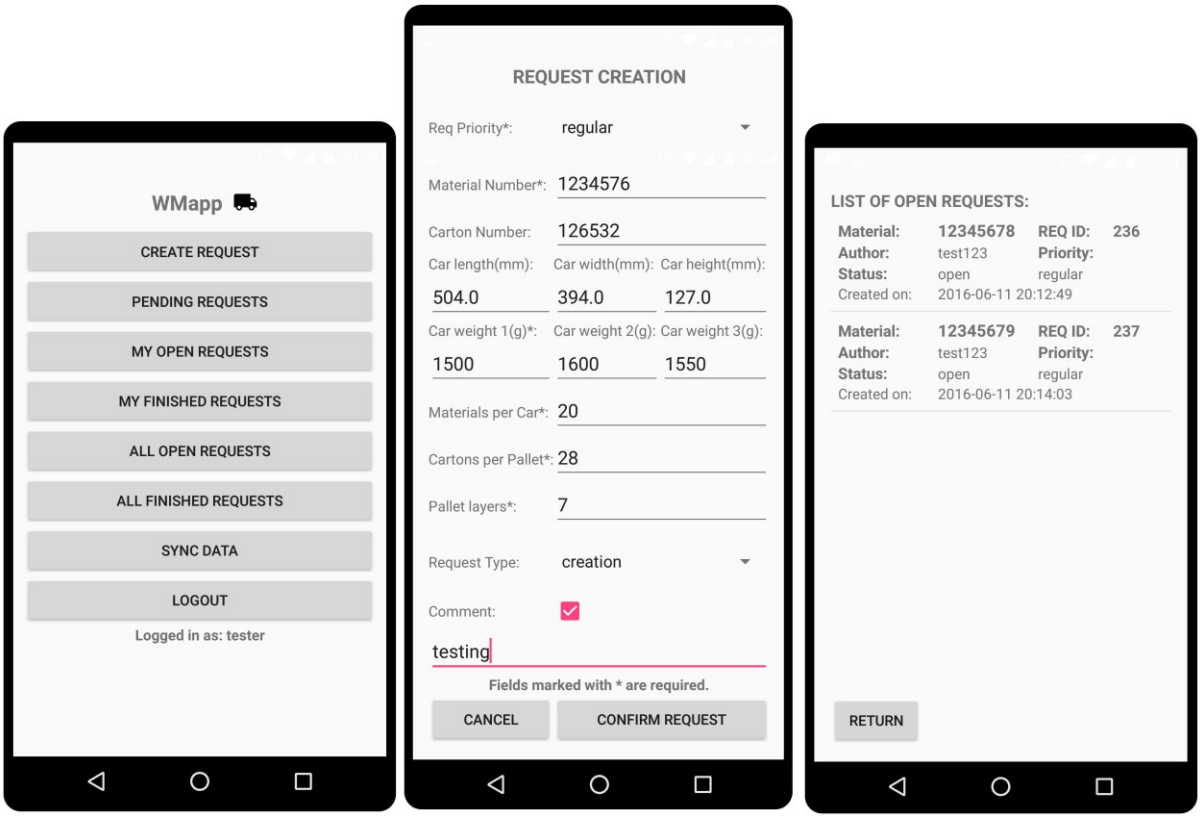
4. Moji zaključeni zahtevki (*My finished requests*): aktivnost, ki izpiše seznam zaključenih zahtevkov uporabnika.

5. Vsi odprti zahtevki (*All open requests*): aktivnost, namenjena skrbnikom osnovnih podatkov, ki izpiše seznam odprtih zahtevkov vseh uporabnikov.

6. Vsi zaključeni zahtevki (*All finished requests*): aktivnost, ki izpiše seznam zaključenih zahtevkov vseh uporabnikov.

7. Sinhronizacija podatkov (*Sync data*): aktivnost, ki služi za prenos določenih podatkov iz strežnika na interno podatkovno bazo naprave SQLite.

8. Odjava (*Logout*): aktivnost, ki izbriše uporabniške podatke iz *SharedPreferences* vmesnika in se vrne na prijavni zaslon.



Slika 5.3: Posnetki aktivnosti aplikacije

Kot lahko opazimo, vsebina zahtevka združuje potrebne vnose standardnega obrazca za popis podatkov pakiranja in Sharepoint zahtevka. Ti se ob potrditvi zabeležijo v tabelo izdelkov in zahtevkov. V kolikor v podatkovni bazi že obstaja material z isto identifikacijsko številko, se njegove specifikacije posodobijo z novimi podatki, drugače pa se izvede nov vnos v tabelo. V sklopu posamezne zahteve je lahko obravnavan zgolj en izdelek. Za isti material je lahko ustvarjenih več zahtevkov, vendar je hkrati v odprtem stanju dovoljen samo eden. S tem se zaščiti sistem pred podvajanjem vnosov. Princip delovanja od druge do šeste aktivnosti je enak, razlikujejo se le po SQL poizvedbah. Vsaka izpiše ustrezen nabor zahtevkov glede na zgoraj opisane kriterije, s pripisano ustrezno informacijo. Ob kliku na željen zahtevek se uporabniku izpiše njegova celotna vsebina. Če ta še ni zaključen, so omogočene določene spremembe, katere, pa je odvisno od uporabniške vloge. Preostale aktivnosti so opisane v nadaljevanju.

5.3.4 Delovanje brez povezave

Znotraj skladišč lahko obstajajo območja, kjer dostop do brezžičnega omrežja ni mogoč. Zato mora aplikacija ob morebitnem izpadu povezave s strežnikom še vedno omogočati delovanje vsaj dveh osnovnih aktivnosti:

- avtentikacijo uporabnika ter
- popis podatkov pakiranja.

Funkcionalnost delovanja sistema brez povezave je bila rešena z vgrajeno Androidovo lokalno podatkovno bazo SQLite. Gre za odprtokodno programsko knjižnico, ki podatke zapisuje neposredno v napravo, zato ne izvaja strežniških operacij. [22]

Vzpostavitev podatkovne baze je enostavna. Tabele se definirajo znotraj funkcije *onCreate()* v Java razredu, razširjenem z metodo *SQLiteOpenHelper*. SQL poizvedbe opredelimo z lastnimi funkcijami, vendar v okviru omenjenega razreda, kot je prikazano v spodnji kodi 5.5.

```
public class DBController extends SQLiteOpenHelper {
        public DBController(Context applicationContext) {
                super(applicationContext, "wmapp.db", null, 1);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
                String query = "CREATE TABLE members ( mName TEXT, mRole TEXT, mUsername
                TEXT UNIQUE, mPassword TEXT, mEmail TEXT UNIQUE, mSalt TEXT, prod_site_id
                INTEGER )";
                db.execSQL(query);
        }
        //....primer lastne funkcije
        public void insertMember(Member member) {
                SQLiteDatabase database = this.getWritableDatabase();
                ContentValues values = new ContentValues();
                values.put("mEmail", member.email);
                values.put("mPassword", member.password);
                //.................
                database.insert("members", null, values);
                database.close();
        }
}
```

Koda 5.5: Primer uporabe SQLite podatkovne baze

Za prvo prijavo v sistem je seveda potrebna povezava s strežnikom, ker na lokalni podatkovni bazi podatki še ne obstajajo. Po uspešnem prvotnem preverjanju pristnosti se podatki uporabniškega računa shranijo v entiteto SQLite podatkovno baze. Ob morebitni odjavi in ponovni prijavi se postopek avtentikacije vrši lokalno. V kolikor gre za novega uporabnika aplikacije na isti napravi, je potrebna ponovna povezava s strežniško podatkovno bazo.

Izpolnjevanje zahtevkov brez povezave je bilo malce težje implementirati, saj se ti morajo v vsakem primeru enkrat poslati na strežnik, sicer jih skrbniki osnovnih podatkov nikoli ne prejmejo in njihova izvršitev ni mogoča. Vgraditi je bilo potrebno logiko, ki zahtevke začasno shrani na lokalno podatkovno bazo naprave in jih ob ponovno vzpostavljeni povezavi s strežnikom avtomatsko pošlje naprej. Takšno funkcionalnost omogoča Androidova metoda *BroadcastReceiver*, ki lahko prestreza različne sistemske signale naprave, med drugim tudi informacijo o povezljivosti z omrežjem. Registracijo ustreznega signala za sprejemnik se nastavi v osnovni XML datoteki aplikacije AndroidManifest.xml. Dokler aplikacije ne zaustavimo, *BroadcastReceiver* čaka v ozadju in ob prejetem signalu sproži lastno funkcijo *onReceive()*. Ta začasno shranjene zahtevke pošlje na strežnik po istem postopku, kot bi se poslali v primeru povezanosti z omrežjem, le da se operacija izvrši v ozadju. Za uporabnike je seznam čakajočih zahtevkov dostopen preko aktivnosti *Pending requests*. Ta med drugim omogoča ročno pošiljanje, v kolikor opisan sprejemnik ni pravilno deloval.

5.3.5 Sistem pošiljanja obvestil

Ena izmed zadnjih implementiranih funkcionalnosti, ki predstavlja dodano vrednost razvite mobilne aplikacije, je sistem pošiljanja obvestil. Kot je že bilo omenjeno, družba za obveščanje o končanih ali ustvarjenih skladiščnih zahtevkih trenutno uporablja elektronsko pošto, ki jo sistem Sharepoint avtomatsko generira in pošlje ustreznim uporabnikom. Medtem ko je za skrbnike osnovnih podatkov tak način obveščanja sprejemljiv, saj njihovo delo večino časa poteka za računalnikom, je za skladiščnike lahko problematičen, saj se v pisarnah ne nahajajo prav pogosto. V urgentnih primerih morajo tako skladiščniki ali skrbniki osnovnih podatkov drug drugega poklicati, da se pravočasno informirajo o stanju zahtev, kar spet prinaša dodatna, lahko tudi zamudna opravila.

Cilj naše aplikacije je torej vzpostaviti način dostavljanja sporočil drugim uporabnikom glede na izvedeno aktivnost. V primeru, da skladiščnik ustvari nov zahtevek, želimo, da so o tem obveščeni vsi skrbniki osnovnih podatkov, saj se do takrat še ne ve, kdo bo zahtevano spremembo izvedel. Ko nekdo iz oddelka za vzdrževanje osnovnih podatkov prevzame zahtevek in spremeni njegov status, mora obveščanje potekati samo še dvosmerno, med avtorjem in moderatorjem spremembe. S tem preprečimo odvečna pošiljanja obvestil, ki bi lahko bila v napoto drugim uporabnikom sistema.

Pametni telefoni z Android platformo imajo vgrajen sistem obveščanja na vrhu zaslona, kjer lahko nameščene aplikacije na napravi puščajo svoja sporočila, tudi če delujejo v ozadju (*notifications*). Zgoraj zahtevano funkcionalnost lahko vgradimo s t. i. Google Cloud Messaging (v nadaljevanju GCM) sistemom. GCM je brezplačna oblačna storitev podjetja Google, ki razvijalcem omogoča implementacijo pošiljanja obvestil iz lastnih strežnikov na naprave Android uporabnikov razvite mobilne aplikacije. Prvi pogoj za delovanje GCM-ja je registracija lastnega projekta na Googlovo konzolo za razvijalce, kjer se generira ustrezni API ključ, s katerim se lahko naš strežnik poveže na storitev GCM. Drugi pogoj pa je nameščena aplikacija Google Play Services na telefonih. Slednja definira identifikacijsko šifro (*token*) naprave, ki ga GCM potrebuje, če želimo sporočila poslati pravemu naslovniku. Ključ se lahko med posodabljanjem naprave spreminja, zato ga moramo shraniti v podatkovno bazo (*membersToken*). Pošiljanje obvestil je potekalo enosmerno, iz strežnika na aplikacije (*Downstream messaging*), kar je zadostovalo potrebam prototipa. Izbran način pošiljanja sporočil je predstavljen v kodi 5.6. Gre za funkcijo s PHP POST metodo, ki se je dodala obstoječim skriptam na strežniku in se proži ob ustreznih dogodkih. Omenjena funkcija generira sporočilo v JSON zapisu, ki se pošlje ustrezni napravi, ta pa ga prestreže z implementirano metodo *GCMListenerService* v aplikaciji. Celoten opis implementacije GCM-ja je dostopen na [23].

```
function generateGCM($role, $req_id, $req_status, $mysqli, $api_key){
        $stmt = $mysqli->prepare("SELECT token FROM membersToken WHERE role = ?");
        $stmt->bind_param('s', $role);
        $stmt->execute();
        $stmt->store_result();
        $stmt->bind_result($token);
        while ($stmt->fetch()) {
                $message = $req_id;
                $url = 'https://android.googleapis.com/gcm/send';
                $fields = array(
                        'registration_ids'  => array($token),
                        'data'                => array( "message" => $message ));
                $headers = array(
                        'Authorization: key=' . $api_key,
                        'Content-Type: application/json');
                $ch = curl_init();
                curl_setopt( $ch, CURLOPT_URL, $url );
                curl_setopt( $ch, CURLOPT_POST, true );
                curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
                curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
                curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );
                $result = curl_exec($ch);
                curl_close($ch);
        }
}
```

Koda 5.6: Funkcija za generiranje GCM sporočil

Zaradi možnosti napak pri delovanju GCM je bilo implementirano tudi generiranje obvestil o stanju zahtevkov preko elektronske pošte, a ker pošiljanje sporočil na službene poštne naslove uporabnikov za testiranje ni bilo odobreno, je bil ta način obveščanja za prototip neuporaben.

5.3.6 Dodatne zahteve uporabnikov

Izdelana mobilna aplikacija je bila nato predstavljena bodočim uporabnikom. Pred pričetkom testiranja smo se posvetovali o morebitnih dodatnih funkcijah, ki bi lahko izboljšale uporabnost sistema. Na željo tako skladiščnikov kot skrbnikov osnovnih podatkov je bila zahtevkom dodana še prioriteta (*request_priorities*), predstavljena v opisu podatkovnega modela. Z njo so lahko skladiščniki definirali nujnost zahtevka, kar Sharepointova stran sicer ne ponuja. Oddelku za vzdrževanje osnovnih podatkov tako pripomore k prednostnem prevzemu urgentnih popravkov v podatkovni bazi.

Skladiščniki so med drugim predlagali, da se polje za vnos komentarja zaradi boljše preglednosti aplikacije prikaže samo po potrebi, saj se v večini primerov ta ne izpolnjuje. Implementirano rešitev lahko vidimo že na sliki 5.3. Ob začetnem odpiranju zahtevka je polje

za komentar skrito. Če ga želimo izpolniti, samo obkljukamo potrditveno okence in prostor za vnos komentarja postane viden uporabniku.

Tretja nadgradnja je bila zahtevnejša. Velikokrat se zgodi, da so prevzemni materiali lastni proizvodi podjetja, ti pa uporabljajo embalažne elemente, ki imajo pakirne podatke že vzdrževane v sistemu, kar smo že opisali pri analizi skladiščnih postopkov. Predlog je bil, da bi se ob vnašanju obstoječe identifikacijske številke kartona ali materiala v sistemu implementirala funkcija za samo-dokončanje (*auto-complete*). Po izboru ustrezne šifre iz podanega nabora, bi logika prenesla obstoječe pakirne informacije elementa neposredno v zahtevek. Nato bi se samo popravili napačni podatki in zahtevek bi bil že izpolnjen. Ker pa je baza obstoječih izdelkov enostavno prevelika in se dnevno povečuje z novimi vnosi, bi tako bila potrebna vgradnja aktivne sinhronizacije SAP-ove podatkovne baze z bazo našega strežnika. To je bilo za prototipni sistem, delujoč zgolj na testnih podatkih, neizvedljivo. Na drugi strani pa je nabor embalažnih elementov manjši in večino časa nespremenljiv, saj je uporaba novih kartonov in s tem potrebnih novih vnosov bolj izjema kot pravilo. Poleg tega so obstoječi pakirni podatki embalaže fiksni in se načeloma ne spreminjajo. Zato je bilo samo-dokončanje in prenos pakirnih podatkov kartonov možno implementirati.

Ker smo želeli obdržati delovanje aplikacije brez povezave, je bilo ob njeni namestitvi najprej potrebno prenesti celoten nabor obstoječih kartonov, shranjenih v tabeli Embalažni elementi (*cartons*), iz strežnika na lokalno podatkovno bazo naprave. Prenos omogoča aktivnost Sinhronizacija podatkov *(Sync data*), ki je opisana v odseku glavnih aktivnostih aplikacije. Funkcija za samo-dokončanje je bila vgrajena s pomočjo Java objektov *AutoCompleteTextView* in *ArrayAdapter*. Njihova implementacija je predstavljena v spodnji kodi 5.7.

```
public class CreateRequest extends AppCompatActivity implements View.OnClickListener {
       //..........
       DBController controller = new DBController(this); //SQLiteOpenHelper razred
       AutoCompleteTextView etCarNumber;
       Integer[] cartons_list;
       @Override
       protected void onCreate(Bundle savedInstanceState) {
              //.............
              etCarNumber= (AutoCompleteTextView) findViewById(R.id.etCartonId);
              cartons_list = controller.getAllCartons();
               ArrayAdapter<Integer> carton_adapter = new
       ArrayAdapter<Integer>(this,android.R.layout.simple_list_item_1,cartons_list);
              etCarNumber.setAdapter(carton_adapter);
       }
       //.....
```

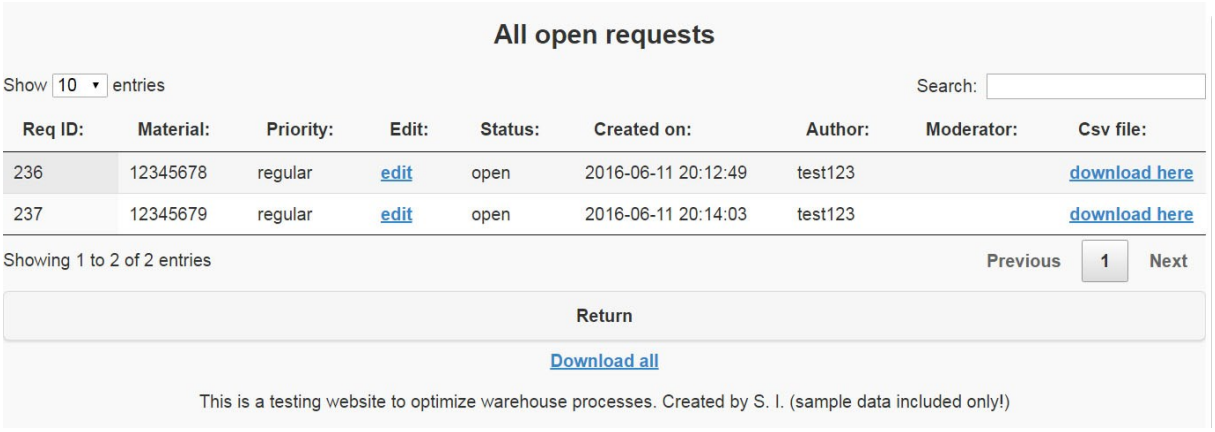
Koda 5.7: Implementacija funkcije za samo-dokončanje

Nato je bilo potrebno omogočiti še prenos pakirnih podatkov neposredno v zahtevek po končanem vnosu izbranega embalažnega elementa. Da bi logika prenesla prave podatke ob pravem času, je bil integriran Androidov vmesnik *OnFocusChangeListener*. Ta se aktivira ob spremembi fokusa trenutnega vnosnega polja. Torej, ko uporabnik spremeni položaj kurzorja iz polja za vnos šifre kartona na drugo vnosno polje na zahtevku, se izvrši prenos pakirnih podatkov izbranega elementa, v kolikor seveda ta obstaja v lokalni podatkovni bazi naprave.

5.4 Razvoj spletnega portala

Zadnja vgrajena komponenta prototipnega sistema je spletni portal, ki omogoča vpogled v bazo obstoječih zahtevkov preko računalnika. Ena izmed osnovnih funkcionalnosti sistema, opisana v četrtem poglavju, namreč zahteva prenos pakirnih podatkov v ustrezni digitalni zapis, ki bi se lahko potencialno uvozil neposredno v interno podatkovno bazo. To bi zelo koristilo skrbnikom osnovnih podatkov. Ker SAP sistem ni nameščen na mobilne naprave, nam prenos podatkov iz aplikacije ne koristi, zato na njej ni bil integriran.

Spletna aplikacija je bila sprogramirana v jeziku HTML in PHP. Tudi tukaj je bilo potrebno preverjanje pristnosti, ki je implementirano z enakim postopkom, kot pri mobilni aplikaciji [20]. Struktura portala je praktično identična razvitemu mobilnemu sistemu, le da poleg vseh funkcij omogoča še prenos pakirnih podatkov v zapisu CSV. Ta se lahko neposredno uvozi v podatkovno bazo preko posebnih transakcij sistema SAP in posodobi pakirne informacije z aktualnimi. Na spodnji sliki 5.5 lahko vidimo izgled spletne strani, ki izpiše nabor vseh trenutno odprtih zahtevkov.



Slika 5.4: Primer uporabe razvitega spletnega portala

Pretvorba podatkov v CSV zapis je bila narejena s PHP funkcijo *fputcsv()*, ki vsako prejeto vrstico iz SQL poizvedbe ustrezno zapiše v nov dokument, ustvarjen s funkcijo *fopen()*, kot je

predstavljeno v kodi 5.10. Datoteka se ob klicu funkcije *fclose()* shrani na ustrezno definirano lokacijo znotraj *header* objekta.

```
<?php
$f = fopen("csvfiles/$mat_id.csv", 'w');
if (!$f) {
        echo "<p>error opening csv!</p>";
} else{
        $result = $con->query("SELECT * FROM materials WHERE mat_id = '$mat_id' LIMIT 1 ");
        while ($row = mysqli_fetch_array($result,MYSQL_ASSOC)) {
                fputcsv($f, array_values($row));
        }
}
fclose($f);
header("Location: /csvfiles/$mat_id.csv");
header("Content-disposition: attachment; filename=$mat_id.csv");
?>
```

Koda 5.8: Zapis CSV datoteke

Grafični izpis seznama zahtevkov je bil implementiran z vtičnikom za jQuery Javascript knjižnico, imenovanim Datatables oz. podatkovne tabele. Gre za zelo prilagodljivo orodje, zasnovano na progresivnih izboljšavah, sposobno dodajanja naprednih interaktivnih kontrol na katero koli HTML tabelo. Med njimi štejemo številčenje strani, dinamično iskanje, več-stolpično razvrščanje ter vrsto drugih dodatkov. [24]

Vgradnja vtičnika je enostavna. Iz omenjenega vira [24] prenesemo željene JS in CSS dokumente, jih naložimo na strežnik, ter jih opredelimo znotraj glave HTML kode. Funkcija, ki definira podatkovno tabelo, je lahko vgrajena znotraj oznake <script> v glavi HTML/PHP skripte ali pa v ločenem JS dokumentu.

5.5 Vpeljava prototipa ter nastale težave

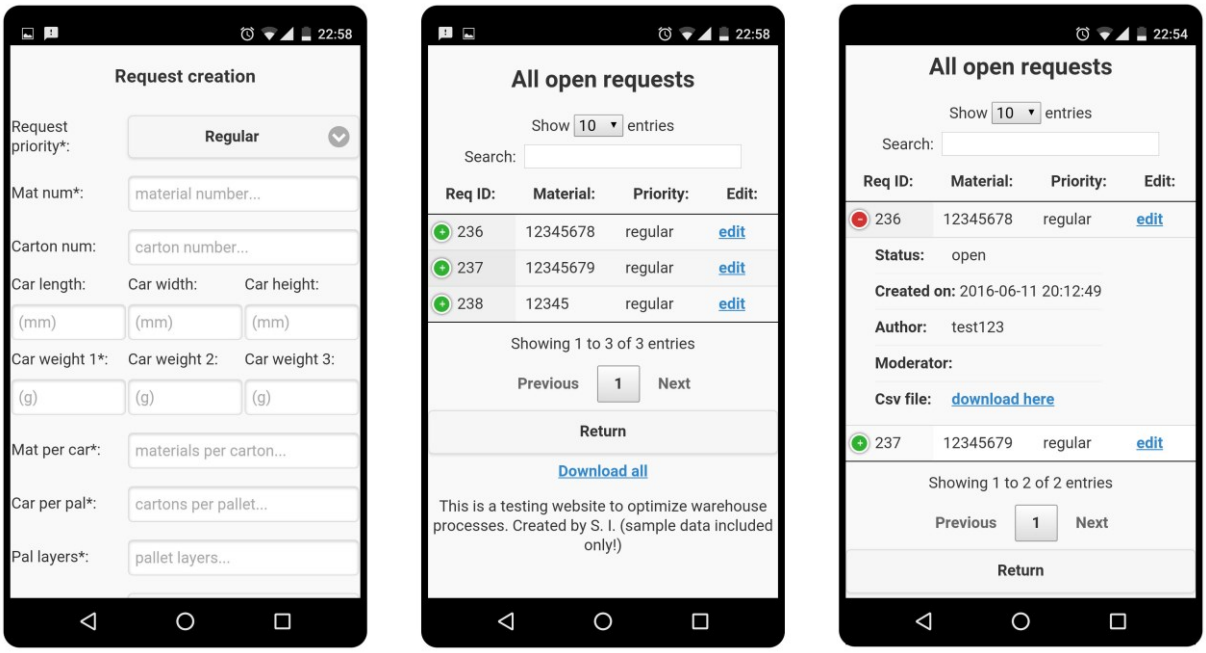
Ko je bil prototipni sistem dokončno izdelan, smo ga lahko vpeljali v realno skladiščno okolje in pričeli z njegovim testiranjem. Seveda pa brez težav ni šlo.

Prvi pereč problem je bil ta, da zaradi varnostnih postopkov podjetja dostop do vzpostavljenega brezžičnega omrežja v skladišču osebnim, neslužbenim mobilnim napravam ni bil odobren oz. bi na odobritev morali čakati več mesecev. To je predstavljalo kar veliko težavo, saj kljub alternativnemu načinu prenosa podatkov preko zunanjih mobilnih omrežij, lahko slednji v zaprtih prostorih za debelimi zidovi zelo slabo deluje. Takšna situacija je bila v depoju na lokaciji B, kjer je tudi potekalo glavno testiranje. Težava je bila odpravljena tako, da se je eden

izmed pametnih telefonov uporabil za prenosno dostopno točko ter bil stacioniran v prostorih, kjer je signal mobilnega omrežja dovolj močan za prenos podatkov. Ker pa je bil oddajajoč signal prenosljivega usmerjevalnika prešibak, smo domet povezave povečali z uporabo ojačevalnika brezžičnih omrežij, ki je z njim segla ravno do kontrolnega prostora, kar je bilo dovolj za uspešno vpeljavo.

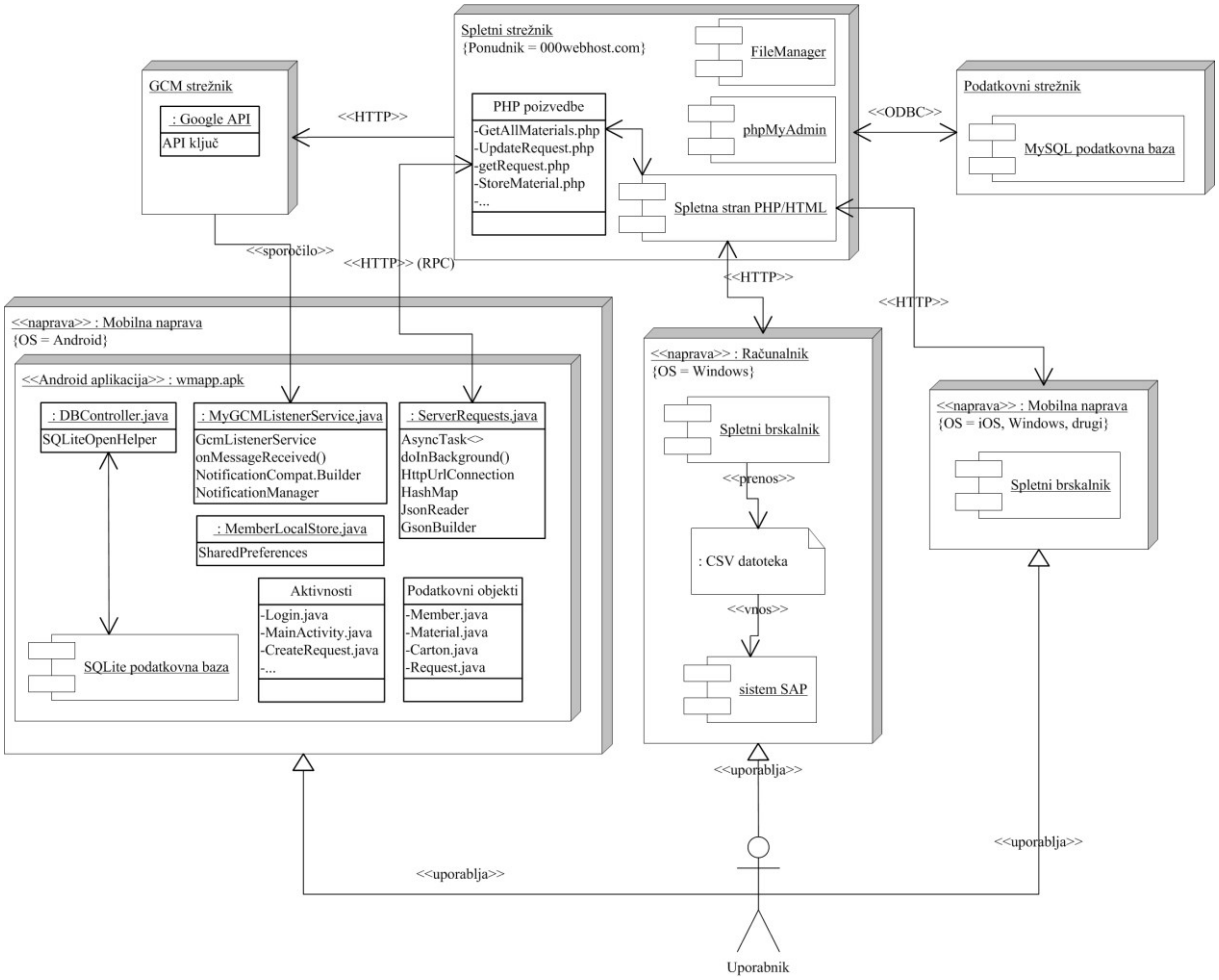
Sodelujoče podjetje nima centraliziranega skladišča, ampak so ti nastanjeni na različnih lokacijah v razdaljah do 30 kilometrov od glavne proizvodnje (slika 2.2). Ker jih je pri testiranju sodelovalo več, bi lahko bila posodobitev aplikacije na telefonih vseh uporabnikov zelo zamudna, zato jo je bilo potrebno naložiti na Googlovo spletno trgovino Google Play. Tako je bilo sodelujočim omogočeno samodejno posodabljanje.

Veliko težavo so predstavljale mobilne naprave. Kljub že prej omenjeni izraziti prevladi Androida na trgu, ima večina zaposlenih sodelujočega podjetja v lasti telefone z Windows ali iPhone OS. Glavni razlog temu je, da družba ne posluje s ponudniki naprav z Googlovo platformo, tako da so vsi službeni telefoni zaposlenih podprti zgolj z omenjenima OS. Da pa bi lahko ne-Android uporabnikom vseeno omogočili preizkus prototipa, se je za rešitev uporabil razviti spletni portal s prilagojenim uporabniškim vmesnikom za mobilne naprave. V obstoječe spletne strani se je tako implementirala knjižnica jQuery Mobile, dostopna na [25], vtičnik Datatables pa je bil nadgrajen z lastnima funkcijama Responsive in RowReorder [24], ki olajšata uporabo pri zaslonih na dotik ter izboljšata preglednost strani z prerazporejanjem vrstic podatkovnih tabel glede na velikost zaslona, kot je prikazano na spodnji sliki 5.5.



Slika 5.5: Spletna aplikacija na mobilni napravi

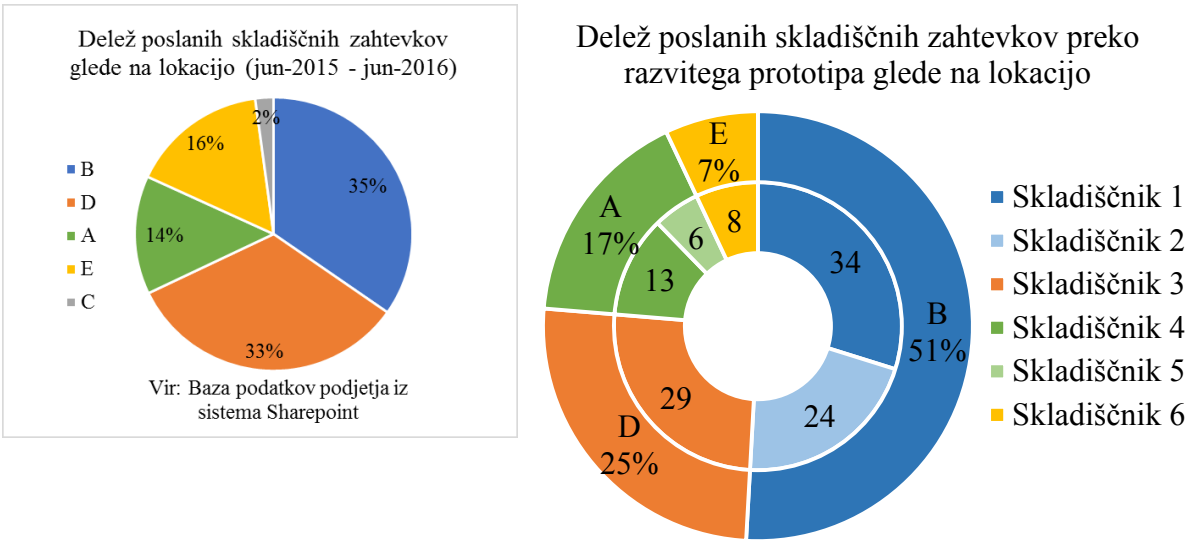
Podrobna zgradba razvitega mobilnega sistema z vsemi predhodno opisanimi komponentami je predstavljena z UML postavitvenim diagramom na sliki 5.6. Ker smo želeli prikazati dele prototipa, s katerimi je uporabnik neposredno v stiku, je bila UML tehnika namerno nekoliko spremenjena. Tako je bil dodan akter (uporabnik), ki lahko v sklopu prototipa uporablja ali mobilno napravo z OS Android ali računalnik ali pa mobilni telefon z drugim OS. Na naprave z OS Android se je namestila razvita mobilna aplikacija, na preostalih pa se je uporabil zgoraj omenjeni spletni portal, dostopen preko obstoječih spletnih brskalnikov. Medtem ko so mobilne telefone uporabljali večinoma skladiščniki za vnose skladiščnih zahtevkov, so na drugi strani skrbniki osnovnih podatkov uporabljali predvsem računalnik. Na njem so imeli nameščen sistem SAP, v katerega so lahko neposredno vnesli dokument CSV z vsemi potrebnimi pakirnimi podatki. Ta se je prenesel iz postavljene spletne strani prototipa. Na diagramu so prikazani tudi ključni deli razvite aplikacije, brez katerih določene predhodno predstavljene komponente ne bi pravilno delovale.



Slika 5.6: UML Postavitveni diagram izdelanega prototipa

Poglavje 6 Rezultati ter analiza testiranja

Testiranje prototipnega sistema je potekalo od 18. aprila 2016 do vključno 20. maja 2016, torej približno en mesec, kar je zadostovalo za potrebno analizo. V tem času je bilo preko razvite aplikacije poslanih 114 posameznih skladiščnih zahtevkov s štirih različnih lokacij (predstavljene na sliki 2.2). Pri raziskavi je sodelovalo šest skladiščnikov in trije skrbniki osnovnih podatkov. Zaradi varstva osebnih podatkov so njihova imena ostala anonimna.



Slika 6.1: Deleži poslanih skladiščnih zahtevkov glede na lokacijo

Na sliki 6.1 so grafično prikazani deleži poslanih zahtevkov s strani skladiščnikov glede na lokacijo depojev, in sicer preko razvitega prototipa v fazi testiranja ter sistema Sharepoint v obdobju enega leta. Iz podanega letnega povprečja izvedenih popravkov pakirnih podatkov ugotovimo, da teh največ prihaja iz najemnih skladišč na lokacijah B in D, ki sta v primerjavi z ostalimi, po kapaciteti in količini skladiščenih končnih izdelkov, največja, pri postopku popisa pa se tudi soočata s precej daljšimi časovnimi izgubami, kar je predstavljeno v nadaljevanju. V raziskavo ni bil vključen depo na lokaciji C, ker se tam skladišči manjša količina surovin, pri katerih pakirni podatki načeloma niso kritični za vzdrževanje v sistemu, kar ponazarja tudi njegov letni delež poslanih zahtevkov. Zaradi pomanjkanja ustreznih mobilnih naprav testiranje prototipa žal ni bilo omogočeno vsem zaposlenim, ki prav tako redno opravljajo obravnavane

postopke v sodelujočih skladiščih. Zgoraj predstavljeni deleži izvedenih zahtev preko razvite aplikacije se zato v nekaterih segmentih razlikujejo od skupnih povprečnih.

Sodelujoča oseba:	Lokacija:	Število poslanih/ obdelanih zahtevkov:	Uporabljena mobilna naprava?	OS mobilne naprave:	Uporabljen računalnik?
Skladiščnik 1	B	34	DA	Android	NE
Skladiščnik 2	B	24	DA	Android	NE
Skladiščnik 3	D	29	DA	Windows	NE
Skladiščnik 4	A	13	DA	Android	NE
Skladiščnik 5	A	6	DA	Windows	NE
Skladiščnik 6	E	8	NE	/	DA
Skrbnik baze 1	A	42	DA	Android	DA
Skrbnik baze 2	A	41	DA	Android	DA
Skrbnik baze 3	A	31	DA	Android	DA

Tabela 6.1: Uporabljene naprave sodelujočih

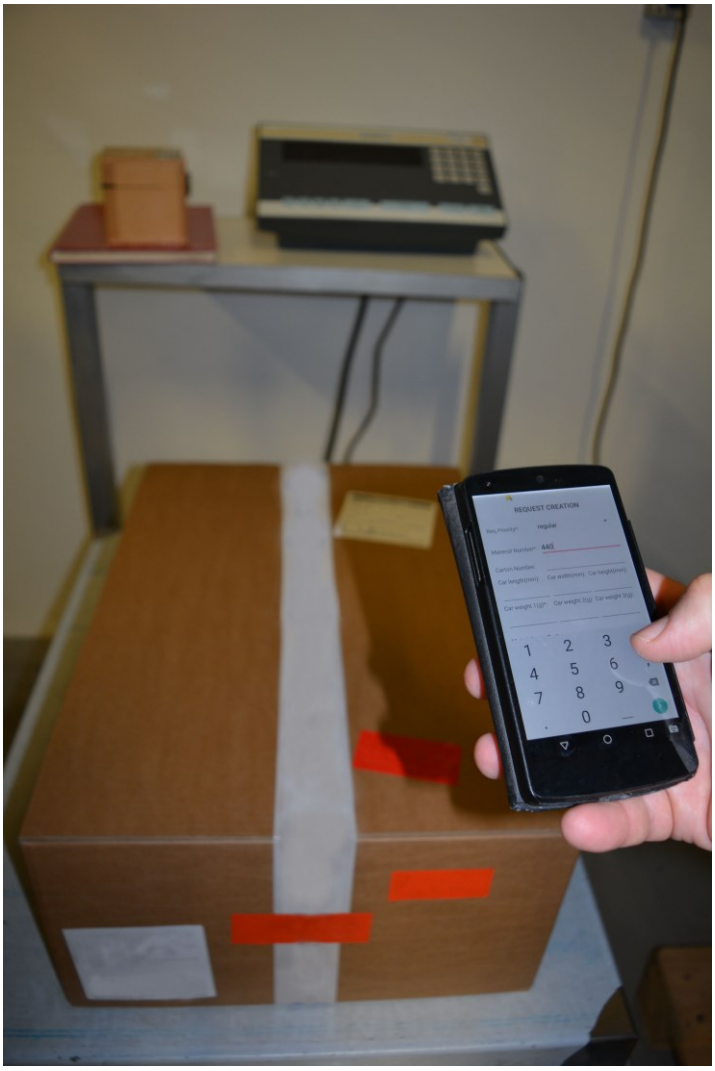
Tabela 6.1 prikazuje uporabljene naprave sodelujočih, njihovo lokacijo ter število poslanih oziroma, s strani skrbnikov baze, obdelanih zahtevkov. Skladiščnik 6 je bil sicer med tistimi, ki nimajo pametnih telefonov, a je kljub temu izrazil željo po testiranju uporabe spletne aplikacije preko računalnika.

V nadaljevanju sledijo primerjave izvedb posameznih operacij pri popisu podatkov pakiranja med trenutnim načinom, kjer se za vnos uporablja papirnati obrazec in opisan sistem Sharepoint, ter vpeljanim novim, kjer se je testiral razvit prototipni sistem. Analizirani časi so bili s strani aplikacije avtomatsko generirani, medtem ko so se za trenutno izvajajoči postopek pridobili z ročnimi meritvami, ki so jih opravile sodelujoče osebe.

6.1 Primerjava časov popisa podatkov pakiranja

Pri prevzemu blaga morajo skladiščniki ob morebitni odkriti napaki vzdrževanih pakirnih podatkov v sistemu iz nabora prejetih palet najprej izbrati tri naključne embalažne elemente in jih pripraviti za meritev, kot je že bilo opisano v poglavju 3. A ker gre za operacijo, s katero se vrši obvezen fizični prenos vzorčnih materialov, ostajata njen način ter čas izvedbe ($t_{priprave}$ na sliki 3.1) z novim postopkom nespremenjena, zato podrobnejša primerjava ni bila potrebna. Enako velja za proces postavitve blaga nazaj na paleto, ki zahteva enako količino časa kot odvzem ($t_{priprave} = t_{postavitve}$).

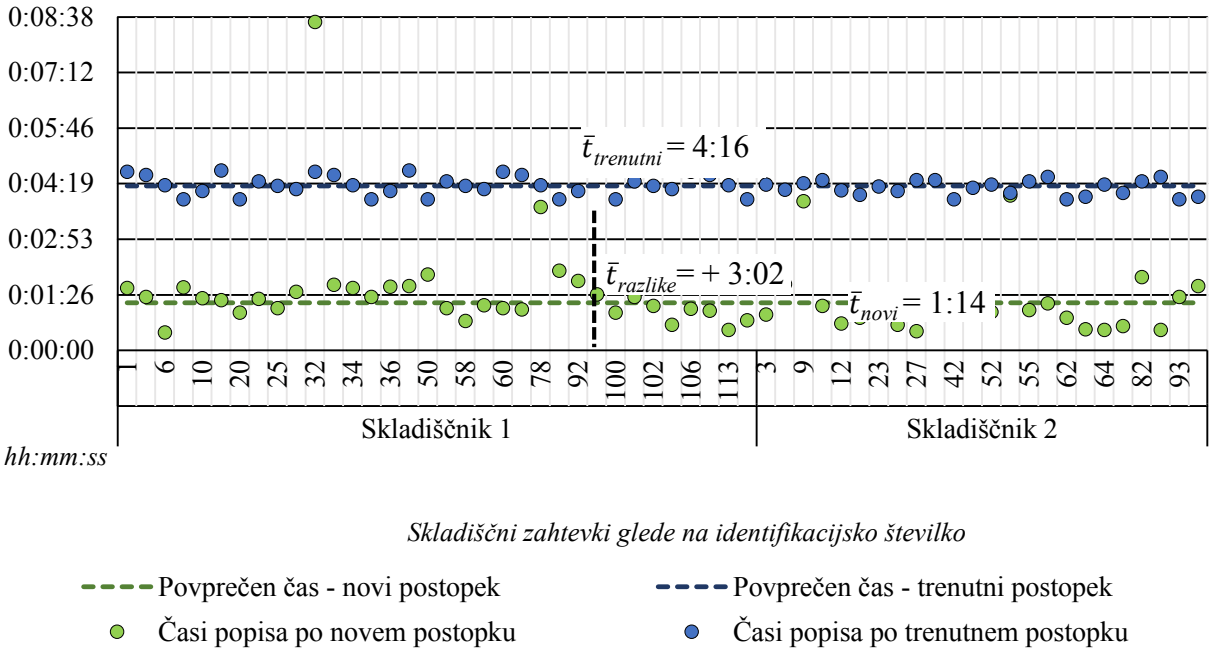
Bistvena sprememba je v načinu popisa podatkov po uspešnih opravljenih meritvah in tehtanju kartonov. V skladu s splošnim postopkom, skladiščniki pridobljene podatke ročno zapišejo na predstavljeno listo iz slike 3.2 oziroma v primeru, da te nimajo pri sebi, najprej na prazen papir, s katerega nato v pisarni prepišejo izmerjene količine na omenjen obrazec. Z novim vpeljanim procesom pa se pakirni podatki vnesejo sočasno z meritvami preko razvite mobilne aplikacije, nameščene na pametnih telefonih uporabnikov (primer uporabe je prikazan na sliki 6.2). Zato smo za enakovredno primerjavo časov popisa pakirnih podatkov($t_{izpolnitve}$ iz definicij časovnih spremenljivk entitete zahtevka), vrednosti časovnih spremenljivk trenutnega postopka, $t_{meritve}$ ter t_{popisa}, ki so jih podali skladiščniki, sešteli. Z njimi smo se že seznanili na sliki 3.1.



Slika 6.2: Prikaz uporabe mobilne aplikacije v skladišču

Omenjena primerjava je v nadaljevanju razdeljena glede na lokacijo skladišč. Prvi diagram na sliki 6.3 tako prikazuje čase popisa pakirnih podatkov po trenutnem in novem postopku na

lokaciji B, kjer je tudi potekalo glavno testiranje s skladiščnikoma 1 in 2. Slednja sta skupaj poslala največ skladiščnih zahtev za popravke v sistemu, obenem pa uporabljala aplikacijo razvito za naprave Android.

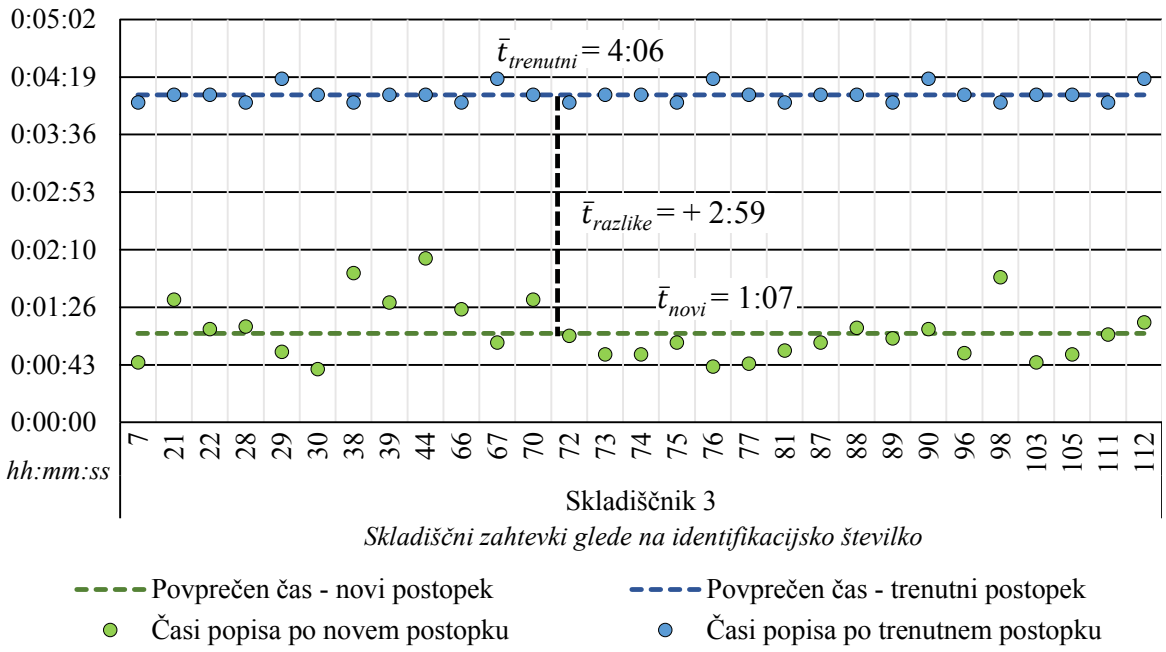


Slika 6.3: Primerjave časov popisa pakirnih podatkov na lokaciji B

V povprečju njun $\bar{t}_{meritve}$ znaša 1:34 (*mm:ss*), $\bar{t}_{popisa}$ = 2:42, skupno torej 4:16 ($\bar{t}_{trenutni}$ na sliki 6.3). Pri vnosih preko mobilne aplikacije pa je kar nekajkrat prišlo do nepričakovanih odstopanj, ki jih lahko vidimo na zgornjem diagramu (na primer zahtevek z identifikacijsko številko 32, ki ga je skladiščnik 1 izdeloval skoraj devet minut). Običajno popis podatkov ob merjenju ne bi smel trajati več kot dve minuti, zato gre najverjetneje za situacijo, ko je uporabnik prototipa že začel izpolnjevati zahtevek, a ga iz določenih razlogov ni uspel takoj dokončati. Ker se prva časovna spremenljivka *open_date* iz opisanega podatkovnega modela zapiše že ob pričetku izdelave, *create_date* pa šele ob potrditvi izpolnjene zahteve, je v tem primeru preračun čas $t_{izpolnitve}$ bistveno daljši od ostalih.

Podobna odstopanja so prisotna tudi pri naslednjih primerjavah, zato se je za vsa povprečja, vključno za ročne popise na papirnatih obrazcih, uporabila modificirana aritmetična sredina. Ta se, za razliko od splošne, izračuna šele po izločitvi skrajnih vrednosti. Za realnejšo analizo smo tako izločili 15 odstotkov izstopajočih podatkov. Tako povprečje popisa izmerjenih količin preko mobilne aplikacije znaša 1:14 ($\bar{t}_{novi}$), kar je za dobre tri minute manj v primerjavi s splošnim postopkom ($\bar{t}_{razlike}$).

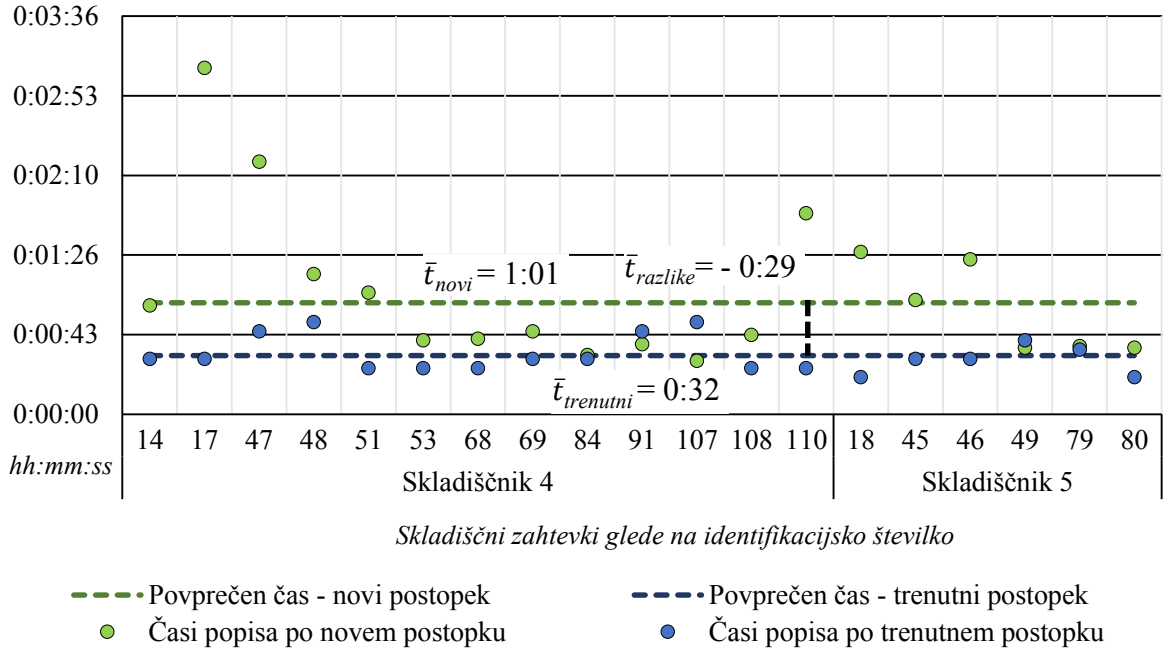
Primerljiva časovna pridobitev je bila z novim načinom pri popisu pakirnih podatkov na strani skladiščne lokacije D, kjer $\bar{t}_{razlike}$ znaša 2:59 (slika 6.4). Na splošno je bil tak rezultat pričakovan, saj, kot smo opisali že prej, gre za depoja, ki skladiščita največjo količino končnih izdelkov. Ti so ključni za prodajo, zato so njihovi pakirni podatki toliko bolj kritični za celovito vzdrževanje, kar pa je s trenutnim procesom precej zamudno.



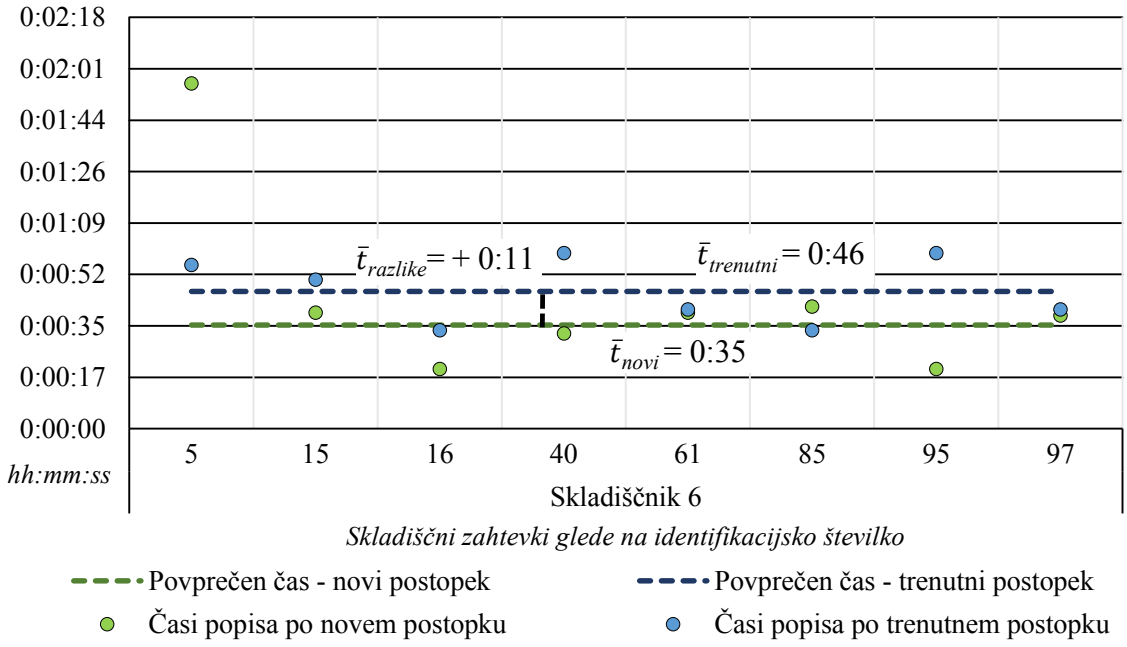
Slika 6.4: Primerjave časov popisa zahtevkov na lokaciji D

Tu velja izpostaviti, da je skladiščnik 3 vnose vršil preko razvite spletne mobilne strani, saj je uporabljal pametni telefon z OS Windows, zato namestitev Android aplikacije ni bila mogoča. Ob tem pa je zanimivo dejstvo, da povprečen čas popisa $\bar{t}_{novi}$ preko spletne strani znaša sedem sekund manj, kljub temu da ima mobilna aplikacija za razliko od nje vgrajeno avtomatsko zaznavo obstoječih kartonov ter vnos njihovih pakirnih podatkov. Razlogov za to je lahko več, od tega da skladiščnik 3 bolje obvladuje uporabo mobilnih naprav, do možnosti, da so se v času testiranja na lokaciji B skladiščili večinoma zunanji izdelki, kjer podatki o uporabljenem kartonu v sistemu ne obstajajo. A ker je bil cilj naloge dokazati vsesplošno uporabnost mobilnih naprav v skladiščnem okolju, se v podrobnejšo analizo tu nismo spuščali.

Ostala sta še zasebna depoja na lokaciji A in E. Primerjava njunih časov je predstavljena na spodnjih diagramih (sliki 6.5 in 6.6).



Slika 6.5: Primerjave časov popisa zahtevkov na lokaciji A



Slika 6.6: Primerjave časov popisa zahtevkov na lokaciji E

Izkaže se, da na lokaciji A (slika 6.5) časovnega prihranka z novim postopkom pri procesu popisa pakirnih podatkov ni, v povprečju gre celo za manjšo izgubo ($\bar{t}_{razlike}$= - 0:29). Pri skladišču na lokaciji E (slika 6.6) je sicer minimalna pridobitev, a je ta v primerjavi s prihranki

v najemnih skladiščih zanemarljiva ($\bar{t}_{razlike}$= + 0:11). Ob temeljiti analizi poteka skladiščenja zasebnih depojev ugotovimo, da ta pakirnega obrazca sploh ne uporabljajo, s čimer prihranijo slabe tri minute. Pri skladišču na lokaciji A je razlog neuporabe v tem, da hranijo večinoma pol-izdelke in surovine, kjer ni potrebno vzdrževati vseh pakirnih podatkov, ki so v sklopu liste za popis. Največkrat so potrebni zgolj popravki tež materialov, zato izpolnitev obrazca ni potrebna. Pri lokaciji E pa je situacija malce drugačna. Skladiščnik 6, ki je sodeloval pri raziskavi, pošilja popravke zgolj za izdelke, kjer je prišlo do spremembe v številu elementov znotraj kartona ali ob uporabi nove embalaže, medtem ko meritev dimenzij in tež ne izvaja, zato so časi vnosa praktično enaki.

Seveda pa je popis pakirnih podatkov zgolj ena izmed izvajajočih se aktivnosti procesa skladiščenja izdelkov, kjer so možni potencialni časovni prihranki. Po odprtju zahtevka tako pride na vrsto sistemska naloga pošiljanja obvestil odgovornim osebam za vzdrževanje pakirnih podatkov v okviru poslanega popravka s strani skladiščnikov.

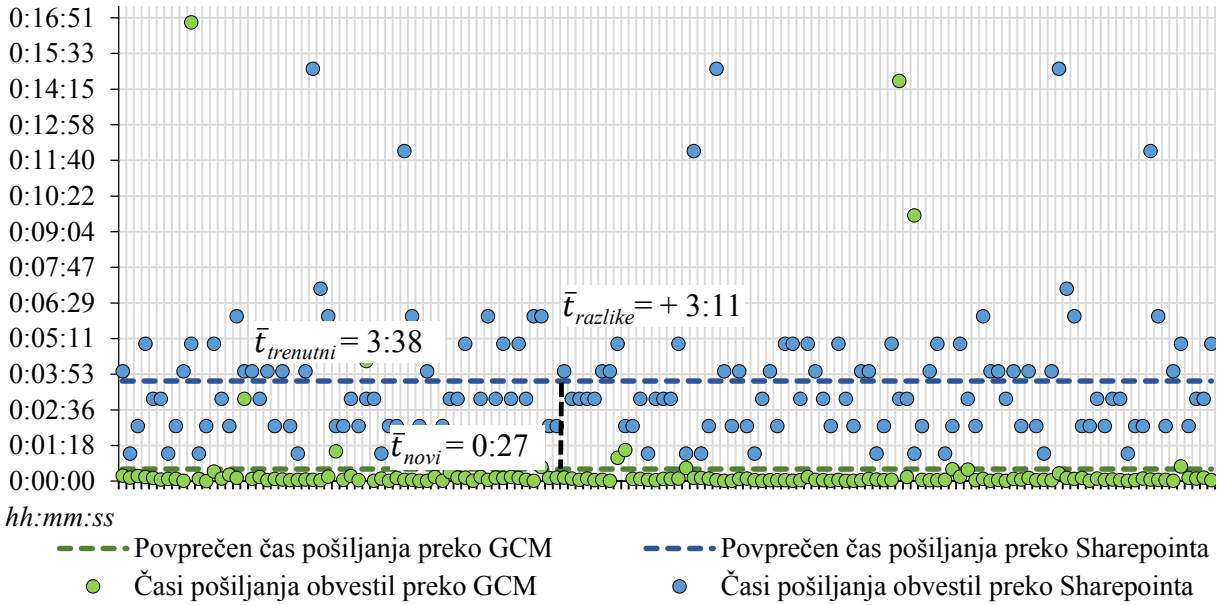
6.2 Analiza posredovanja zahtevkov na strežnik

Prototipni sistem je beležil tudi čase posredovanja skladiščnih zahtevkov iz mobilne naprave na strežnik ($t_{posredovanja}$ iz petega poglavja). Rezultati uporabnikov Android telefonov kažejo, da je bilo takojšnje prejetje ($t_{posredovanja}$ = 0:00) 80,2 odstotno uspešno. Za preostale njihove popravke pa je sistem v povprečju potreboval $\bar{t}_{posredovanja}$ = 3:18. Najverjetnejši razlog je seveda v izpadu omrežja oziroma nahajanju skladiščnikov izven dosega brezžične povezave. Vnosi preko spletne aplikacije so bili pri posredovanju podatkov na strežnik sicer 100 odstotno uspešni, a to zgolj zaradi tega, ker začasno shranjevanje neposredno na napravo sploh ni bilo omogočeno, torej se je v primeru prekinjene povezave zahtevek v celoti izbrisal.

6.3 Primerjava časov pošiljanja sistemskih obvestil

Zaposleni so večkrat omenjali težave pravočasnega prejemanja obvestil o stanju odprtih zahtevkov s strani Sharepointa. Ker naj bi sistemsko pošiljanje v primeru urgenc trajalo bistveno predolgo, mora komunikacija med oddelki steči preko telefonskih klicev. To prinaša dodatna usklajevanja in nepotrebno časovno izgubo. Problematiko potrjujejo tudi rezultati raziskave. Ti so bili za trenuten postopek skladiščenja pridobljeni iz nabora 98 poslanih skladiščnih zahtevkov sistema Sharepoint. Ta beleži v svoji podatkovni bazi točen datum in čas odprtja zahtev, ki je bil nato odštet od časa prejete elektronske pošte. Tako dobimo ustrezne vrednosti $t_{obvestila}$.

Na drugi strani pa razvit prototip obvestila generira z GCM sistemom, opisanem v petem poglavju, kjer je tudi predstavljen postopek izračuna časov $t_{obvestila1}$ in $t_{obvestila2}$. Ker je sam namen pošiljanja obvestil, po katerem se spremenljivki tudi razlikujeta, za analizo nepomemben in ker so vrednosti obeh med seboj enakovredne, so bile pri diagramu na sliki 6.7 združene v en sam niz. Ta prikazuje primerjavo pridobljenih časov pošiljanja s strani obeh sistemov.



Slika 6.7: Primerjava časov pošiljanja sistemskih obvestil

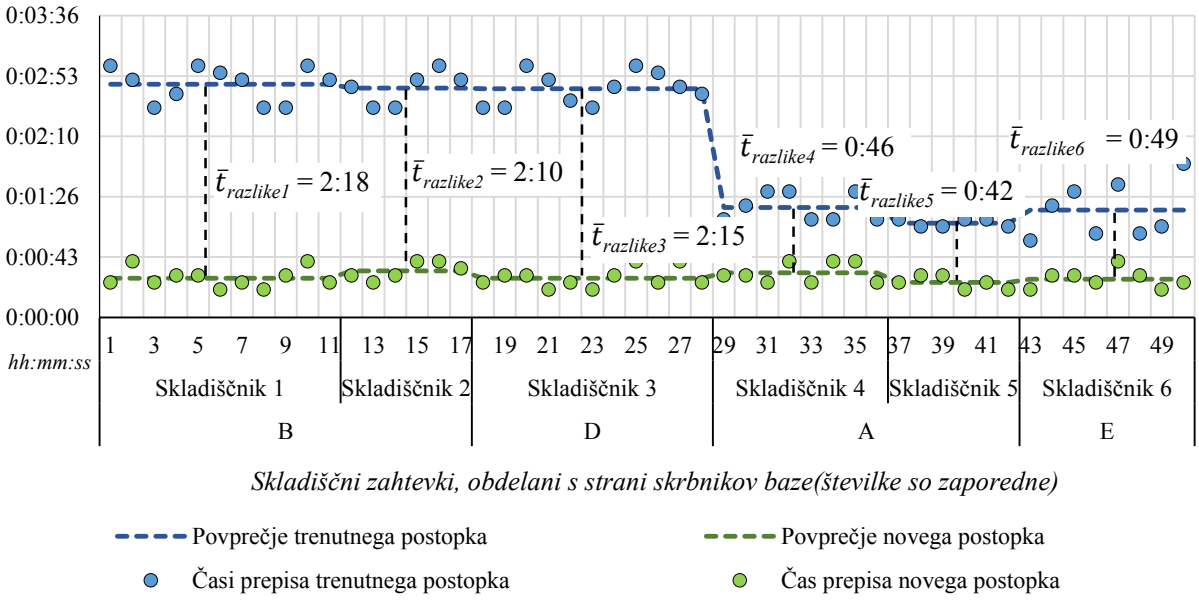
Ugotovimo, da Sharepoint za uspešno posredovanje obvestil povprečno porabi $\bar{t}_{trenutni}$ = 3:38, kar je za dobre tri minute več kot pošiljanje preko GCM-ja ($\bar{t}_{novi}$ = 0:27). Tukaj je sicer potrebno povedati, da večja časovna odstopanja, dolga tudi po več ur, ki so se pojavila s strani prototipa, na diagramu niso prikazana, saj bi ta bil zaradi prevelikih časovnih razlik nepregleden. Za računanje povprečij smo ponovno uporabili modificirano aritmetično sredino, s katero so bili omenjeni odstopi prav tako izločeni. Do teh primerov je prišlo, kadar telefoni skladiščnikov ali skrbnikov dalj časa niso bili povezani z omrežjem. Razlogi so lahko zaustavljeni podatkovni prenosi ob koncu dela, prazna baterija, nahajanje skladiščnikov izven dosega povezljivosti ipd. Opisanih odstopov zato tudi nismo analizirali, saj gre za zunanji vzrok in ne napako sistema.

6.4 Časovne pridobitve s strani skrbnikov baze

Ko skrbniki osnovnih podatkov prejmejo informacijo o novo odprtem skladiščnem zahtevku, morajo iz slednjega najprej prepisati izmerjene količine in jih pripraviti za vzdrževanje. Šele nato sledijo popravki v podatkovni bazi. Pakirni podatki se po splošnem postopku nahajajo v

pripeti PDF kopiji izpolnjenega obrazca oziroma za skladišča, ki ga ne uporabljajo, neposredno v tekstovnem polju zahtevka, namenjenega za komentar. Ker sta oba načina prepisa, predvsem prvi, precej zamudna, se je v okviru prototipa omogočil prenos posamičnih zahtevkov v CSV datoteki, predstavljenem v petem poglavju. Omenjen dokument je tudi primeren za neposreden vnos podatkov v podatkovno bazo.

Pridobljeni časi prepisa so bili ročno izmerjeni s strani skrbnikov baze. Ker večjih odstopanj ni bilo, je bilo dovolj pregledno združiti rezultate vseh lokacij na en diagram (slika 6.8). Za trenutni postopek so skrbniki osnovnih podatkov vzeli stare zahtevke sodelujočih skladiščnikov ter ponovno opravili potrebne prepise ter pripravo podatkov, razen pri novo prispelih zahtevah istih uporabnikov, ki so omogočale meritve pri tekočem delu.



Slika 6.8: Primerjava časov prepisa pakirnih podatkov iz zahtevka

Hitro opazimo, da je za prepis podatkov pri zahtevkih skladiščnikov 1, 2 in 3 s trenutnim postopkom potrebnega precej več časa ($\bar{t}_{trenutni1,2,3}$ = 2:45), saj tudi redno uporabljajo pakirne obrazce, medtem ko so povprečni časi obdelave popravkov skladiščnikov 4, 5 in 6 v primerjavi s prvimi tremi za več kot pol krajši ($\bar{t}_{trenutni4,5,6}$ = 1:15). Z novim postopkom je čas prepisa vedno med 25 in 40 sekund, ne glede na izvor zahtevka. Tako pri najemnih skladiščih povprečno pridobimo dobri dve minuti, pri zasebnih pa okoli 46 sekund.

Čas vzdrževanja ($t_{popravka}$), za katerega smo predstavili izračun pri definiciji časovnih spremenljivk našega podatkovnega modela, žal ni bilo možno pravilno sistemsko beležiti. Čeprav so se sodelujoči iz oddelka za vzdrževanje osnovnih podatkov držali dogovora in po

svojem opravljenem delu spremenili status zahtevkov v *in process* ali v primeru brez globalnih popravkov neposredno v status *finished*, je omenjen izračun kljub temu netočen. Ta sicer predstavlja razliko med časom spremenjenega statusa in prejetega obvestila, kar je načeloma prav. A izkaže se, da skrbniki redko kdaj pričnejo s popravki pakirnih podatkov takoj po prispelem sporočilu, saj opravljanje skladiščnih zahtev ni njihova edina obveznost. Tako v povprečju potrebujejo več ur za dokončanje zahtevkov, seveda pa se ta čas razlikuje glede na njihovo prioriteto in v katerem delu dneva so bili poslani.

Zaradi tega je bilo potrebno čase vzdrževanja s strani skrbnikov baze ročno izmeriti in oceniti. Iz pridobljenih časov ugotovimo, da je proces vnosa podatkov, potem ko je priprava in obdelava zaključena, enak ne glede na izvor, zato se ga z novim postopkom tudi ni dalo bolje optimizirati.

6.5 Primerjava celovite izvedbe skladiščnih zahtevkov

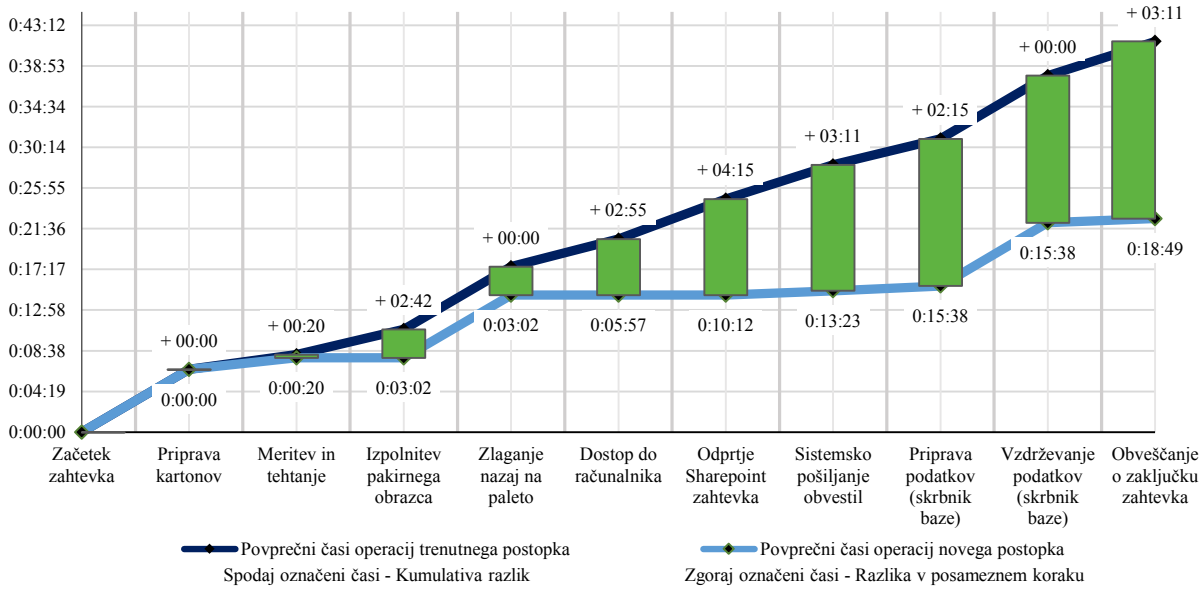
Ko skrbniki končajo s svojimi obveznostmi, je zahtevek uradno zaključen. Sistem mora samo še obvestiti skladiščnika o uspešni izvedbi popravka.

Seveda pa ne smemo pozabiti na ključno dodano vrednost uporabe mobilne tehnologije - to je dostopnost naprav. Namreč vsakič, ko se izvede popis pakirnih podatkov, morajo skladiščniki pakirni obrazec ali list papirja, izpolnjenega z izmerjenimi količinami, odnesti do prostora, kjer je na voljo računalniška oprema. Temu sledi kopiranje liste za popis, v kolikor se jo uporabi, prijava na računalnik, vzpostavitev s sistemom Sharepointom ter odprtje zahtevka znotraj njega. Vse te aktivnosti zaposlenim v depojih vzamejo ogromno časa, ki pa so z mobilno aplikacijo uspešno odpravljene.

Zato je bilo za končno analizo smiselno, da zgoraj omenjeno pridobitev predstavimo v sklopu primerjave celovite izvedbe skladiščnih zahtevkov med trenutnim in novim vpeljanim procesom. S tem dobimo tudi boljši vpogled na časovne prihranke. Diagrami, ki sledijo, vsebujejo časovnico zaporednega izvajanja vseh operacij, potrebnih za uspešno integracijo skladiščnih popravkov v podatkovni sistem. Primerjave smo znova razdelili glede na lokacije, saj so pridobljeni povprečni časi skladiščnikov iz istih depojev praktično identični. Vrednosti nad »padajočimi palicami« predstavljajo časovne razlike izvedb posameznih aktivnosti med trenutnim in novim načinom, vrednosti pod njimi pa skupno absolutno časovno razliko do tedaj izvršenih procedur.

Diagrame smo razvrstili na enak način kot prej. Tako so na sliki 6.9 najprej prikazani rezultati iz lokacije B, pri kateri so časovne pridobitve tudi največje. Tu z novim postopkom ni prihrankov zgolj pri treh operacijah: pripravi kartonov, zlaganju nazaj na paleto in vzdrževanju

podatkov, ki so že maksimalno optimirani. Izboljšave smo pri sistemskem pošiljanju obvestil ter popisu pakirnih podatkov že opisali, s tem da je slednji na spodnjih diagramih predstavljen z dvema podoperacijama. To sta meritev ter izpolnjevanje pakirnega obrazca, ki smo ju pri prejšnjih primerjavah zaradi boljše preglednosti združili v eno. Čase vnosa podatkov pri prototipnem sistemu smo podali v sklopu procedure meritev in tehtanj, saj vnosi v aplikacijo z njo sočasno potekajo. Tako nam ostaneta še dve aktivnosti – dostop do računalnika in odprtje Sharepoint zahtevka. Skladiščnika 1 in 2 od kontrolnega prostora do pisarne v povprečju potrebujeta $\bar{t}_{dostopa}$ = 2:55, kar uporaba mobilnih naprav, kot že povedano, povsem odpravi. Odpravljena pa je tudi časovna izguba druge operacije, odprtje Sharepoint zahtevka, ki je med vsemi aktivnostmi najbolj potratna, saj skladiščnika v povprečju z njo izgubita dobre štiri minute ($\bar{t}_{prenosa}$ = 4:15). Ta predstavlja skupek večih nalog, od kopiranja izpolnjenega obrazca, prijave na računalnik, prenosa PDF dokumenta nanj, do vzpostavitve s sistemom Sharepoint, kjer se zahteva kreira. Zato tudi traja tako dolgo.

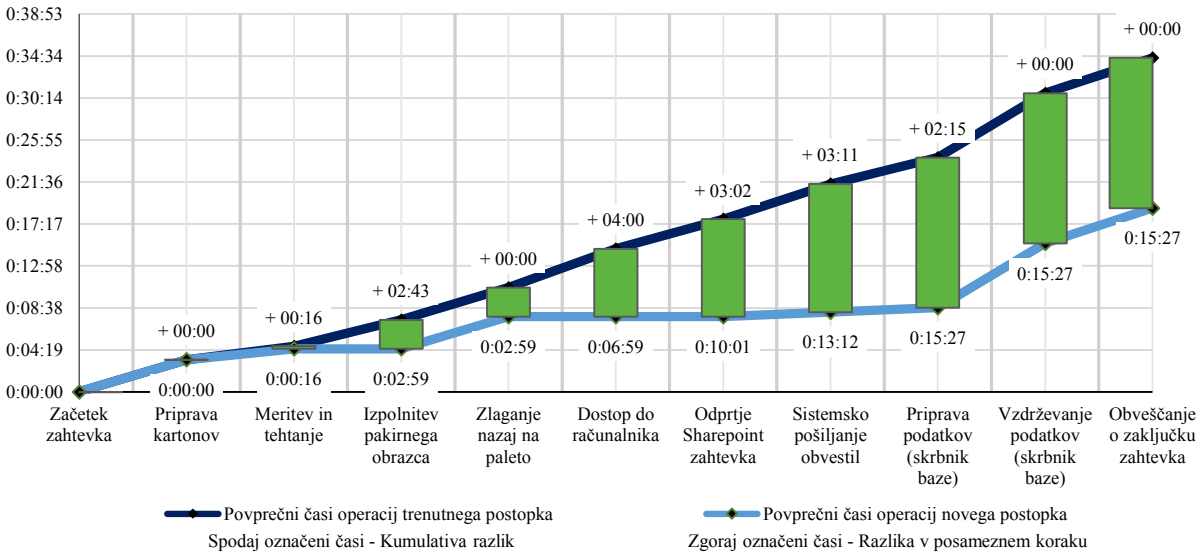


Slika 6.9: Primerjava celovite izvedbe zahtevkov na lokaciji B

V celoti z novim vpeljanim postopkom na B lokaciji v povprečju prihranimo skoraj 19 minut na zahtevek, kar je 45 odstotkov manj v primerjavi s trenutnim načinom izvedbe. Od tega skladiščnika 13:23, skrbniki baze pa 5:26. Če upoštevamo, da je bilo v 23 delovnih dneh, kolikor je trajalo testiranje, iz lokacije B poslanih 58 popravkov, skladiščnika z uporabo mobilne tehnologije dnevno pridobita približno 34 minut, skupno pa skoraj 13 ur na mesec. Na drugi strani, skrbniki osnovnih podatkov dnevno prihranijo malo manj kot 14 minut, skupno torej več kot dve uri na mesec. Če rezultata obeh strani združimo, samo na lokaciji B mesečno

prihranimo dobra dva delovna dneva, kar letno znaša kar cel delovni mesec. Seveda pa se frekvenca pošiljanja zahtevkov tako na dnevnem, mesečnem kot tudi letnem nivoju zelo spreminja. Precej je odvisna od stanja na trgu, zahtev končnih kupcev glede načina dostave produkta, zaradi katerih do napak v podatkovni bazi sploh prihaja, časa dopustov, določenih sprememb v poslovanju ter drugih dejavnikov, na katere ne moremo vplivati. Zato je tudi težko oceniti realni letni prihranek podjetja pri vpeljavi novega načina izvajanja skladiščnih procesov, ki ga zaradi zgoraj omenjenih razlogov ter manjšega deleža vseh dejansko poslanih zahtevkov na preostalih lokacijah nismo podali.

Na sliki 6.10 je predstavljena celovita analiza drugega najemnega depoja z lokacije D. V primerjavi s prejšnjim sta dve bistveni razliki. Prva je čas dostopa do računalnika, ki je še za minuto daljši ($\bar{t}_{dostopa}$ = 4:00), saj se pisarna z ustrezno računalniško opremo nahaja čisto na drugi strani skladišča kot odpremna in prevzemna točka. Druga razlika pa je pri sistemskem obveščanju uporabnikov o stanju zahtevkov. Skladiščnik 3 je namreč za testiranje prototipnega sistema uporabljal telefon z Windows OS. GCM sporočila so se sicer lahko generirala, tako da so skrbniki baze še vedno prejemali informacijo o novih popravkih, medtem ko skladiščnik 3 obvestila o zaključku zahteve neposredno preko mobilne naprave žal ni mogel prejeti. Časovna izguba zadnje aktivnosti tako ni odpravljena.

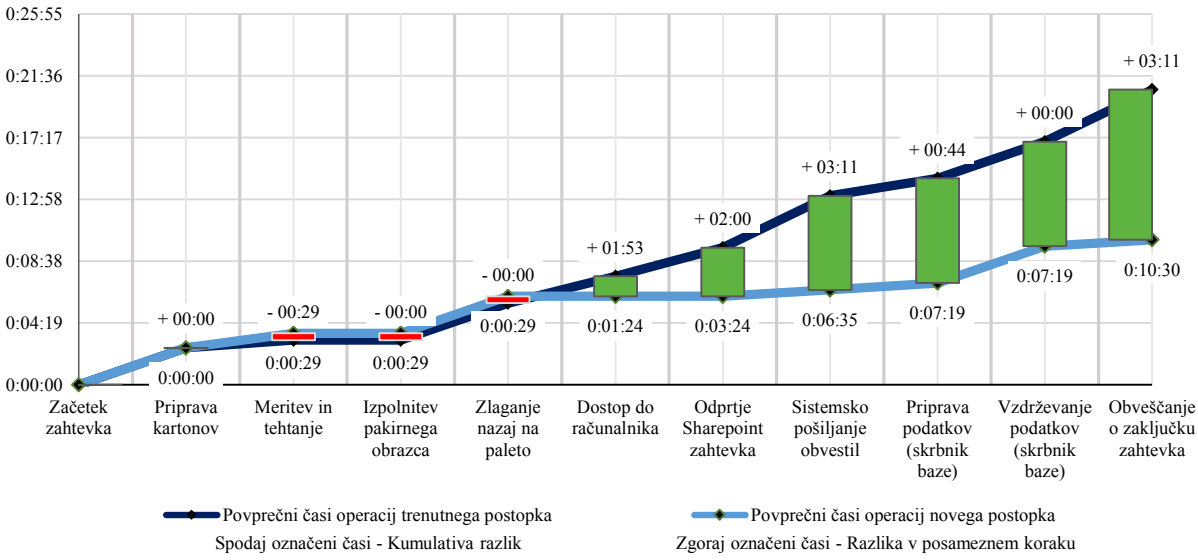


Slika 6.10: Primerjava celovite izvedbe zahtevkov na lokaciji D

Še vedno pa celoten prihranek znaša več kot 15 minut, kar je sicer količinsko manj kot na lokaciji B, a še vedno s 45 odstotno izboljšavo glede na trenutno izvedbo skladiščnih popravkov

v sklopu lokacije D, kar je ogromno. Skladiščnik 3 od tega pridobi 10:01, skrbniki pa ponovno 5:26.

Nadaljujemo z zasebnimi skladišči. Kot smo že opisali, pride pri popisu pakirnih podatkov na lokaciji A do manjše izgube, kar lahko opazimo tudi na sliki 6.11. Z novim postopkom tako skladiščnika 5 in 6 najbolj prihranita pri odpravi dostopa do računalniške opreme ($\bar{t}_{dostopa}$ = 1:53) ter odprtja Sharepoint zahteve ($\bar{t}_{prenosa}$ = 2:00). Čeprav skladiščnik 6 ni imel nameščene Android aplikacije iz enakih razlogov kot skladiščnik 3, smo vseeno upoštevali časovni prihranek zaključnega sistemskega obveščanja s strani skladiščnika 5, se pravi z uporabo GCM pošiljanja.

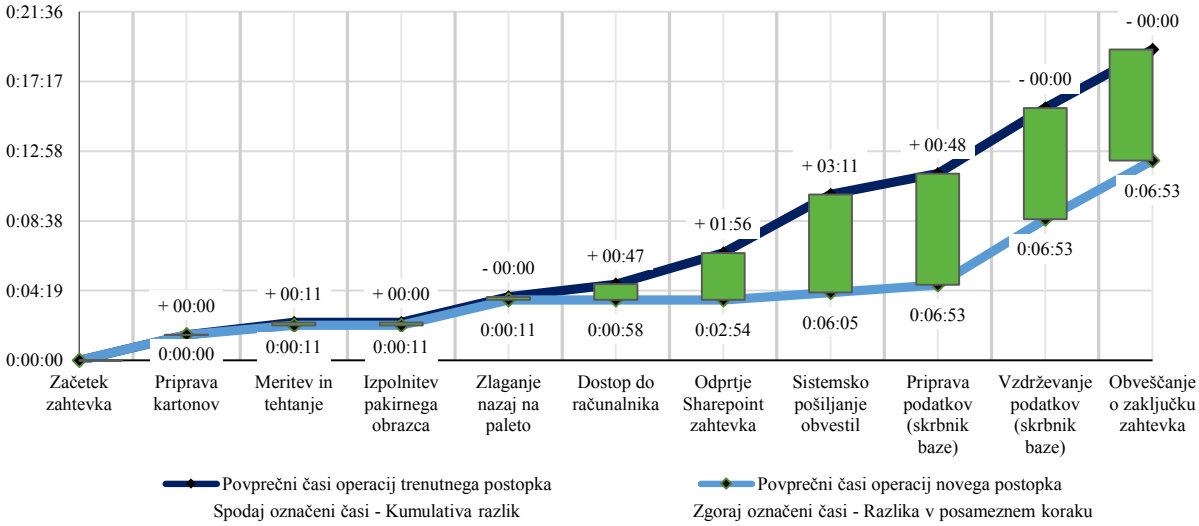


Slika 6.11: Primerjava celovite izvedbe zahtevkov na lokaciji A

Ker popravki, ki prihajajo iz zasebnih skladišč, niso tako kompleksni kot pri najemnih, je tudi celotni čas izvedbe tako z novim kot trenutnim postopkom bistveno krajši. Kljub temu da so pri popisu na lokaciji A beležene manjše časovne izgube, pa s prototipom vseeno skupno prihranimo kar 50 odstotkov izvedbenega časa oziroma 10:30, od tega skladiščnika 6:35, skrbniki pa 3:55.

Ker je bil cilj naloge predstaviti uporabnost mobilne tehnologije, smo pri celotni izvedbi novega postopka na lokaciji E (slika 6.12) upoštevali, kot da so bili zahtevki poslani preko spletne aplikacije na mobilni napravi, čeprav je skladiščnik 6 ni imel. Omenjeno aplikacijo je sicer uporabljal, a na računalniku. Dostop do slednjega tako potencialno znova odpravimo ($\bar{t}_{dostopa}$ = 0:47) kot tudi vzpostavitev s sistemom Sharepoint ($\bar{t}_{prenosa}$ = 1:56). Časovni prihranki na strani

skladiščnika 6 so z uporabo prototipa izmed vseh lokacij tu najmanjši in sicer 2:46, medtem ko skrbniki baze pridobijo 3:59. Skupaj torej znašajo približno sedem minut, kar je za 35 odstotkov manj kot z običajnim postopkom. Razloga za slabši izkupiček sta dva. Prvi je čas dostopa do računalnika, ki je v primerjavi z ostalimi najkrajši ($\bar{t}_{dostopa}$ = 0:47). Tega so zaposleni izboljšali s postavitvijo opreme sredi depoja. Drugi pa je v uporabi spletne aplikacije namesto mobilne, ki tako kot pri skladiščniku 3 iz lokacije D, nima možnosti sprejemanja GCM sporočil.



Slika 6.12: Primerjava celovite izvedbe zahtevkov na lokaciji E

Poglavje 7 Sklepne ugotovitve

Ob zaključku testiranja sta vodji skladišč ter skrbnikov osnovnih podatkov podala strokovno mnenje o razvitem prototipnem mobilnem sistemu ter novem načinu izvajanja skladiščnih procesov. Oba sta se strinjala, da so pozitivne lastnosti nove aplikacije sledeče:

- zanesljivost (pošlje ustrezne pakirne podatke),
- uporabnost (časovno skrajša izvedbo potrebnih operacij),
- skladnost z uporabniškimi zahtevami,
- nizki stroški implementacije (združljiva z obstoječo tehnološko opremo zaposlenih),
- prijaznost do uporabnikov (enostaven dizajn ter osnovni meni) ter
- možnost uporabe obstoječega omrežja.

Vodja skladiščnikov je med drugim izpostavil še prednosti uporabe mobilnih naprav na mestu izvajanja skladiščnih operacij, ki odpravi potrebne dostope do računalniške opreme, avtomatskega prejemanja obvestil o izvedenih sistemskih popravkih, ki omogočajo takojšnje nadaljevanje skladiščnih procesov ter potencialno znižanje stroškov s prenehanjem uporabe papirnatih obrazcev.

Vodja skrbnikov osnovnih podatkov, na drugi strani, pa je poudaril izboljšavo njihovega trenutnega procesa z dodano funkcijo prenosa pakirnih podatkov v zapis, primeren za hiter, neposreden vnos v obstoječi podatkovni sistem podjetja. Glede na to, da delo njegove ekipe večinoma poteka za računalnikom in ima glavno dodano vrednost prototipna razvita spletna aplikacija, vidi prednost uporabe mobilnih naprav predvsem pri urgentnih primerih ob odsotnosti celotnega oddelka, saj komunikacija med njimi in skladiščem tako steče bistveno hitreje. Medtem ko pri splošni uporabi predlaga raje vgraditev pošiljanja obvestil preko elektronske pošte, ki jo tudi redno pregledujejo, v kolikor so seveda možni isti časovni prihranki kot z GCM obveščanjem.

Pomisleke pa sta imela glede Android naprav, za katere je bila aplikacija prvotno tudi razvita. Izpostavila sta, da so bile vse zasebne, zato uporabo s strani zaposlenih sodelujoče podjetje ne more zahtevati. Ker se zaenkrat razpolaga samo s službenimi telefoni, ki imajo OS Windows ali iOS, bi družba morala razmisliti o možnosti novega sodelovanja s ponudniki telefonov sistema Android ali pa bilo potrebno obstoječo aplikacijo prilagoditi drugim OS, ob

predpostavki, da bi še vedno omogočala enake časovne prihranke. Druga težava, ki sta jo izpostavila, je uporaba mobilnega omrežja. Kjer brezžična povezava z interno mrežo v depojih ni mogoča, bi se skladiščniki morali s telefonom prijaviti na mobilno omrežje. To bi povzročilo dodatne stroške, ki bi jih morali nekateri zaposleni zaradi individualno sklenjenih pogodb kriti sami. Zadnji, tretji problem pa vidita v varnosti podatkov, saj je razvit prototip zaradi potrebnih povezav z zunanjim spletnim strežnikom izpostavljen dodatnim neželenim vdorom. Zaščita poslovnih informacij je za družbo kritična, zato bi ta morala biti vzpostavljena na višjem nivoju ter obvezno usklajena z odgovornimi iz IT oddelka podjetja, ki so tudi zadolženi za odobritev vpeljave novih informacijskih sistemov. Na tem mestu se predlaga celo uporaba internega strežnika, s čimer bi potencialno odpravili potrebo po zunanjih dostopih, v kolikor seveda postavljeno brezžično omrežje znotraj skladišč deluje na vseh kritičnih točkah, kar pa zaradi neodobrenega dostopa v fazi testiranja ni bilo mogoče preveriti.

Vodji sta po pogovoru s svojimi zaposlenimi, ki so sodelovali pri preizkusu prototipa, ob koncu sklenili, da je nov način izvajanja skladiščnih procesov praktičen in učinkovit. Ob določenih prilagoditvah in nadaljnjem razvoju sistema, ki bi uspešno odpravila zgoraj omenjene težave, pa vidita možnost tudi v dejanski produkcijski implementaciji.

Glede na izpeljano raziskavo ter podano končno analizo lahko povzamemo, da je bil osnovni cilj diplomskega dela, s katerim smo želeli predstaviti in dokazati uporabnost mobilne tehnologije v realnem skladiščnem okolju, dosežen. S prenosljivostjo pametnih naprav ter odzivnim komunikacijskim sistemom, razvitim na podlagi poznavanj poslovnih procesov sodelujočega podjetja, smo uspešno optimizirali skladiščne postopke, končni rezultati testiranja pa so potrdili časovne prihranke v različnih segmentih uporabe novega produkta.

Opravljena študija je tudi dobra osnova za produkcijski razvoj skladiščnih mobilnih aplikacij, saj predstavi ključne izzive in naloge, s katerimi se mora za uspešno implementacijo mobilnih sistemov v obstoječe depoje soočiti razvijalec. Ob tem pa je potrebno poudariti, da izpostavljene prednosti uporabljene tehnologije v sklopu diplomske naloge, kot sta dostopnost in učinkovita komunikacija, še zdaleč niso edine dodane vrednosti mobilnih naprav. Številne funkcije, ki imajo že danes vgrajene, fotoaparat, GPS, NFC ter druge, odpirajo možnosti različnih izboljšav in nadgradenj, ki so lahko v poslovnem okolju uporabne. S tehnološkimi napredki in spremembami, ki nas v prihodnosti še čakajo, pa lahko pričakujemo, da bo teh še mnogo več.

Literatura

[1] C. S. Patil, R. R. Karhe in M. A. Aher, »Development of Mobile Technology: A survey«, *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 1, št. 5, str. 374, november 2012.

[2] GSMA. (2016). *GSMA Mobile Economy 2016* [Online]. str. 10, 16. Dosegljivo: https://www.gsmaintelligence.com/research/?file=97928efe09cdba2864cdcf1ad1a2f58c&download. [Dostopano: 15. 6. 2016].

[3] Cecil C. Bozarth in Robert B. Handfield, *Introduction to operation and supply chain management*, 2. izd. New Jersey: Pearson Prentice Hall, 2008.

[4] P. B. Schary in T. Skjøtt-Larsen, *Managing the Global Supply Chain*. Copenhagen: Copenhagen Business Press, 1995.

[5] David Frederick Ross, *Distribution: Planning and Control, Managing In The Era Of Supply Chain Management*, 2. izd. Dordrecht: Kluwer Academic Publishers, 2004.

[6] D. M. Lambert, James R. Rock in Lisa M. Ellram, »Warehounsing,« v *Fundemantals of Logistics Management*. Boston: MCGraw-Hill, 1998, str. 266-268, 276-278.

[7] Definicija Sharepointa. *Podporne strani Microsoft Office* [Online]. Dosegljivo: https://support.office.com/en-us/article/What-is-SharePoint-97b915e6-651b-43b2-827d-fb25777f446f. [Dostopano: 15. 6. 2016].

[8] D. Loshin, »Master Data and Master Data Management,« v *Master Data Management*. New York: Morgan Kaufmann OMG Press, 2009, str. 9.

[9] SAP [Online]. Dosegljivo: http://go.sap.com/corporate/en/company.faqs.html. [Dostopano: 15. 6. 2016].

[10] Warehouse OS [Online]. Dosegljivo: http://www.warehousemobilesolutions.com/. [Dostopano: 15. 6. 2016].

[11] Snappii [Online]. Dosegljivo: https://www.snappii.com/resource-center/warehouse-management-app/. [Dostopano: 15. 6. 2016].

[12] Rfgen [Online]. Dosegljivo: http://www.rfgen.com/sap-enterprise-mobility#sap-mobile-applications. [Dostopano: 15. 6. 2016].

[13] 000webhost [Online]. Dosegljivo: https://www.000webhost.com/. [Dostopano: 15. 6. 2016].

[14] Android OS [Online]. Dosegljivo: https://www.android.com/. [Dostopano: 15. 6. 2016].

[15] Gartner. (maj 2016). *Global market share held by smartphone operating systems* [Online]. Dosegljivo: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. [Dostopano: 15. 6. 2016].

[16] Android Developers, *Connecting to the Network* [Online]. Dosegljivo: https://developer.android.com/training/basics/network-ops/connecting.html. [Dostopano: 15. 6. 2016].

[17] JSON [Online]. Dosegljivo: http://www.json.org. [Dostopano: 15. 6. 2016].

[18] GSON [Online]. Dosegljivo: https://github.com/google/gson. [Dostopano: 15. 6. 2016].

[19] Secure Hash Standard (SHS), FIPS PUB 180-4, avgust 2015.

[20] WikiHow, *How to Create a Secure Login Script in PHP and MySQL* [Online]. Dosegljivo: http://www.wikihow.com/Create-a-Secure-Login-Script-in-PHP-and-MySQL. [Dostopano: 15. 6. 2016].

[21] Android Developers, *SharedPreferences* [Online]. Dosegljivo: https://developer.android.com/reference/android/content/SharedPreferences.html. [Dostopano: 15. 6. 2016].

[22] SQLite [Online]. Dosegljivo: https://www.sqlite.org/about.html. [Dostopano: 15. 6. 2016].

[23] Android Developers. (maj 2016). *Try Cloud Messaging for Android* [Online]. Dosegljivo: https://developers.google.com/cloud-messaging/android/start. [Dostopano: 15. 6. 2016].

[24] Datatables [Online]. Dosegljivo: https://datatables.net/. [Dostopano: 15. 6. 2016].

[25] JQuery Mobile [Online]. Dosegljivo: http://jquerymobile.com/. [Dostopano: 15. 6. 2016].